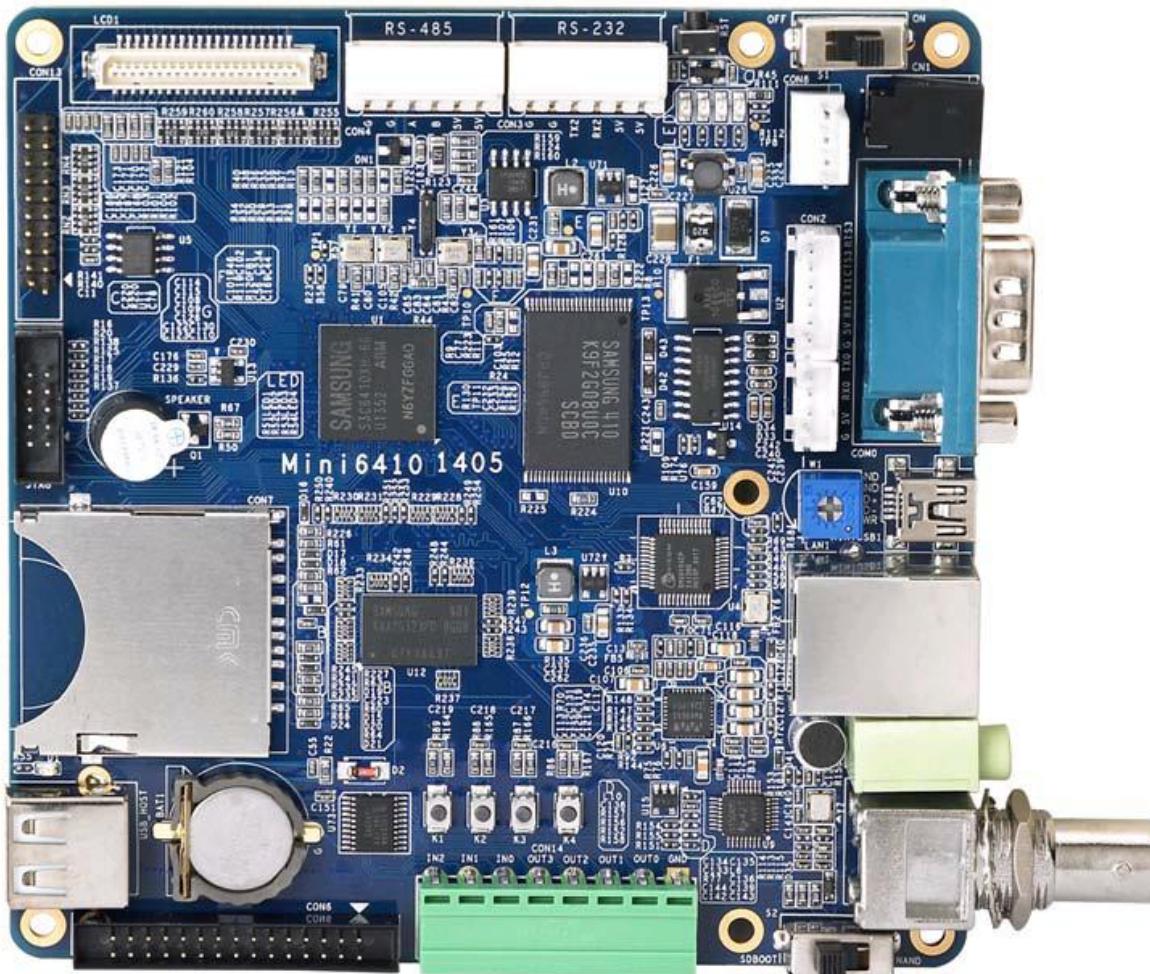


Mini6410-1405 User's Manual



| REVISION | ORIGINATOR | SCR | REV DATE |
|----------|-----------------------|-----|--------------|
| 0.1.1 | Friendly ARM Co., Ltd | | July 8, 2015 |
| | | | |
| | | | |

Friendly ARM 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 Friendly ARM Co., Ltd.

**Friendly ARM Co., Ltd
Guangzhou, China**

copyright@2010

COPYRIGHT STATEMENT

The content (content being images, text, programs and scripts) of this English manual is copyright © Friendly ARM Co., Ltd. All rights expressly reserved.

Any content of the manual printed or downloaded may not be sold, licensed, transferred, copied or reproduced in whole or in part in any manner or in or on any media to any person without the prior written consent of Friendly ARM Co., Ltd including but not limited to:

- transmission by any method
- storage in any medium, system or program
- display in any form
- performance
- hire, lease, rental or loan

Requests for permission to reproduce material from this manual should be addressed to Friendly ARM Co., Ltd.

Index

| | |
|---|----|
| Chapter 1 Introduction | 9 |
| 1.1 Introduction to Mini6410-1405..... | 10 |
| 1.1.1 Mini6410-1405 Overview | 10 |
| 1.1.2 Mini6410-1405 Hardware Features..... | 11 |
| 1.1.3 Board Dimension | 13 |
| 1.1.4 Linux Features | 14 |
| 1.1.5 WinCE 6.0 Features..... | 18 |
| 1.1.6 Android Features | 20 |
| 1.2 Board Schematics | 21 |
| 1.2.1 Jumper | 21 |
| 1.2.2 Schematic..... | 21 |
| 1.3 Interface Specifications | 22 |
| 1.3.1 Address Space | 22 |
| 1.3.2 Power | 23 |
| 1.3.3 Serial Port..... | 23 |
| 1.3.4 USB Interface | 24 |
| 1.3.5 Network Interface..... | 25 |
| 1.3.6 Audio Interface | 25 |
| 1.3.7 AV-IN..... | 26 |
| 1.3.8 JTAG Interface | 26 |
| 1.3.9 LED | 28 |
| 1.3.10 User Key | 28 |
| 1.3.11 LCD Interface | 29 |
| 1.3.12 ADC Input | 30 |
| 1.3.13 PWM Buzzer | 30 |
| 1.3.14 I2C-EEPROM | 31 |
| 1.3.15 SD Card | 31 |
| 1.3.16 SDIO-II//SD-WiFi..... | 32 |
| 1.3.17 GPIO | 33 |
| Chapter 2 Get Started | 34 |
| 2.1 System Setup and Configuration..... | 34 |
| 2.1.1. Boot Options..... | 34 |
| 2.1.2 Connect Peripherals | 34 |
| 2.1.3 Set Up Hyper Terminal | 35 |
| 2.2 Burn BIOS to SD Card | 35 |
| 2.2.1 Introduction to BIOS..... | 36 |
| 2.2.2 Burning BIOS in WindowsXP | 38 |
| 2.2.3 Burn BIOS in Windows 7 | 40 |



| | |
|--|----|
| 2.3 Play with Superboot | 44 |
| 2.3.1 Install Windows CE..... | 45 |
| 2.3.2 Install Android..... | 46 |
| 2.3.3 Restore Linux..... | 46 |
| 2.3.5 FriendlyARM.ini..... | 47 |
| 2.4 Introduction to Supertool..... | 50 |
| 2.4.1 Install USB Download Driver | 51 |
| 2.4.2 Superboot's Menu..... | 57 |
| Chapter 3 Install, Upgrade and Run Systems..... | 59 |
| 3.1 Install Systems with Minitools | 59 |
| 3.1.1 Install Minitools | 60 |
| 3.1.1.1 Install on Windows | 60 |
| 3.1.1.2 Install on Linux | 61 |
| 3.1.2 Flash Superboot to SD Card | 61 |
| 3.1.3 Install Systems with Minitools..... | 62 |
| 3.2 Install System from SD Card | 66 |
| 3.2.1 Install Linux (YAFFS2)..... | 66 |
| 3.2.2 Install WindowsCE6..... | 67 |
| 3.2.3 Install Android (YAFFS2)..... | 68 |
| Chapter 4 Explore Linux | 69 |
| 4.1 Get Started with Qtopia-2.2.0 and QtE-4.8.5 | 69 |
| 4.1.1 Calibrate Touch Screen | 70 |
| 4.1.2 Main Window | 71 |
| 4.1.3 SMPlayer | 72 |
| 4.1.3.1 Play Video with SMPlayer | 74 |
| 4.1.3.2 Convert Video Files | 76 |
| 4.1.4 WiFi AP | 80 |
| 4.1.5 Play MP3 | 82 |
| 4.1.6 Play Video..... | 83 |
| 4.1.7 View Pictures..... | 84 |
| 4.1.8 Auto Mount SD Card/Flash Drive..... | 84 |
| 4.1.9 Calculator | 86 |
| 4.1.10 Terminal..... | 86 |
| 4.1.11 File Manager | 87 |
| 4.1.12 Ethernet Configuration | 88 |
| 4.1.13 Setup WiFi..... | 89 |
| 4.1.13.1 Launch WiFi Utility | 90 |
| 4.1.13.2 Connect to Wireless AP | 90 |
| 4.1.13.3 Disconnect WiFi | 94 |
| 4.1.13.4 Configure WiFi IP | 94 |



| | |
|--|-----|
| 4.1.14 Ping Test | 96 |
| 4.1.15 Web Browser | 97 |
| 4.1.16 LED Test..... | 97 |
| 4.1.17 EEPROM..... | 98 |
| 4.1.18 PWM Buzzer..... | 99 |
| 4.1.19 Serial Port Assistant | 100 |
| 4.1.20 Connect to Internet via GPRS Modem | 104 |
| 4.1.21 Messaging via GPRS Modem | 107 |
| 4.1.22 3G Dial-Up | 110 |
| 4.1.23 Audio Recording..... | 115 |
| 4.1.24 USB Camera..... | 117 |
| 4.1.25 Preview with Camera | 117 |
| 4.1.26 LCD Test..... | 118 |
| 4.1.27 Backlight Control..... | 119 |
| 4.1.28 A/D Conversion..... | 120 |
| 4.1.29 User Button Test..... | 121 |
| 4.1.30 Touch Pen Test | 122 |
| 4.1.31 Barcode Scanner | 122 |
| 4.1.32 Lauguage Setting | 123 |
| 4.1.33 Set Up Time Zone, Date, Time and Alarm Clock..... | 124 |
| 4.1.34 Screen Rotation..... | 126 |
| 4.1.35 Setup Auto Run Program..... | 127 |
| 4.1.36 System Shutdown..... | 129 |
| 4.1.37 Watchdog..... | 130 |
| 4.1.37 Start QtE-4.8.5 | 131 |
| 4.1.39 Which Qt to Choose..... | 133 |
| 4.2 Play Mini6410-1405 via Hyper Terminal | 133 |
| 4.2.1 Play MP3 | 134 |
| 4.2.2 Terminate Program | 135 |
| 4.2.3 Mount USB Drive/Portable Hard Disk..... | 135 |
| 4.2.4 Mount SD Card..... | 137 |
| 4.2.5 File Transfer To and From PC via Serial Port | 138 |
| 4.2.6 LED Test..... | 140 |
| 4.2.7 User Button Test | 142 |
| 4.2.8 Serial Port Assistant | 142 |
| 4.2.9 PWM Buzzer | 144 |
| 4.2.10 Backlight Control..... | 144 |
| 4.2.11 I2C-EEPROM Test | 145 |
| 4.2.12 AD Conversion | 146 |
| 4.2.13 USB WiFi/SD WiFi..... | 146 |



| | |
|---|-----|
| 4.2.14 Telnet | 155 |
| 4.2.15 Ethernet Configuration | 156 |
| 4.2.16 Set MAC Address | 158 |
| 4.2.17 Telnet Mini6410-1405 | 159 |
| 4.2.18 FTP | 160 |
| 4.2.19 Manipulate LED via WEB | 161 |
| 4.2.20 Mount NFS | 162 |
| 4.2.21 Set System Clock | 162 |
| 4.2.22 Save Data to Nand Flash | 163 |
| 4.2.23 Set Up Auto Run Program | 163 |
| 4.2.24 Take Screenshots with Snapshot | 165 |
| 4.2.25 Check RAM Info | 165 |
| 4.3 Set up Fedora 9.0 Development Environment | 167 |
| 4.3.1 Install Fedora 9.0 | 167 |
| 4.3.2 Add User Account | 181 |
| 4.3.3 Access Windows Files | 183 |
| 4.3.4 Set up NFS Service | 188 |
| 4.3.5 Set up Cross Compile Environment | 194 |
| 4.4 Uncompress Source Code and Install Application Utilities | 196 |
| 4.4.1 Uncompress Source Code | 196 |
| 4.4.2 Create Target File System | 200 |
| 4.4.3 Install Target File System | 201 |
| 4.4.4 Install LogoMaker | 202 |
| 4.5 Compile U-Boot | 203 |
| 4.5.1 Compute U-Boot That Supports NAND Booting | 204 |
| 4.5.2 Compile U-Boot That Supports SD Card Booting | 204 |
| 4.6 Compile Kernel | 205 |
| 4.6.1 Compile Kernel | 205 |
| 4.6.2 Drivers | 206 |
| 4.7 Compile Busybox | 208 |
| 4.8 Make File System Image | 209 |
| 4.8.1 Make YAFFS2 Image | 209 |
| 4.8.2 Make UBIFS Image | 209 |
| 4.8.3 Make EXT3 Image | 210 |
| 4.9 Sample Linux Programs | 210 |
| 4.9.1 “Hello, World” | 211 |
| 4.9.2 LED Test Program | 215 |
| 4.9.3 User Button Test Program | 216 |
| 4.9.4 PWM Buzzer | 218 |
| 4.9.5 I2C-EEPROM Program | 221 |



| | |
|--|------------|
| 4.9.6 Serial Port Program | 224 |
| 4.9.7 UDP Program..... | 229 |
| 4.9.8 Application of Math Libraries..... | 233 |
| 4.9.9 Thread Programming..... | 233 |
| 4.9.10 Pipe Programming – Manipulate LED via WEB..... | 235 |
| 4.9.11 “Hello, World” C++ Program | 239 |
| 4.10 Sample Linux Driver Programs | 241 |
| 4.10.1 Hello Module..... | 241 |
| 4.10.2 LED Driver..... | 246 |
| 4.10.3 User Button Driver | 248 |
| 4.11 Compile Qtopia-2.2.0..... | 253 |
| 4.11.1 Uncompress and Install Source Code..... | 253 |
| 4.11.2 Compile and Run Qtopia-2.2.0 for X86 | 253 |
| 4.11.3 Compile and Run Qtopia-2.2.0 for ARM | 255 |
| 4.12 Compile QtE-4.8.5 | 257 |
| 4.12.1 Uncompress and Install Source Code | 257 |
| 4.12.2 Compile and Run QtE-4.8.5 for ARM | 257 |
| Chapter 5 Explore WinCE6 | 260 |
| 5.1 Get Started with CE6 | 260 |
| 5.1.1 Calibrate Touch Screen | 260 |
| 5.1.2 Screen Rotation..... | 262 |
| 5.1.3 Check System Info | 263 |
| 5.1.4 Set Time Zone and Date | 263 |
| 5.1.5 User Data | 264 |
| 5.1.6 Mount Flash Drive/SD Card..... | 264 |
| 5.1.7 Play MP3 | 265 |
| 5.1.8 LED Test..... | 266 |
| 5.1.9 User Button Test | 266 |
| 5.1.10 PWM Buzzer Test..... | 267 |
| 5.1.11 Audio Recording | 267 |
| 5.1.12 Serial Port Assistant | 268 |
| 5.1.13 “Hard Decoding” Player | 270 |
| 5.1.14 Set MAC Address | 273 |
| 5.1.15 Configure Ethernet..... | 275 |
| 5.1.16 SD WiFi | 276 |
| 5.1.17 USB WiFi..... | 278 |
| 5.1.18 USB Bluetooth..... | 278 |
| 5.1.19 Backlight Control | 281 |
| 5.1.20 Watchdog..... | 282 |
| 5.1.21 Synchronize with PC (Windows 7)..... | 283 |



| | |
|---|-----|
| 5.2 Set up Windows CE6 Development Environment | 287 |
| 5.2.1 Install Visual Studio 2005 and Patches | 290 |
| 5.2.2 Install WindowsCE 6.0 and Patches | 300 |
| 5.2.3 Install Tencent QQ (A Third Party Messenger) | 316 |
| 5.2.4 Install BSP and Sample Programs | 319 |
| 5.3 Configure and Compile WinCE6.0 Kernel and Bootloader | 322 |
| 5.3.1 Compile Kernel | 322 |
| 5.3.2 Configure LCD Type and Serial Port in BSP | 328 |
| 5.3.3 One Wire Precise Touching | 329 |
| 5.3.4 Bootloader | 331 |
| 5.3.5 Create SDK | 335 |
| 5.3.6 Install SDK | 336 |
| Chapter 6 Explore Android | 341 |
| 6.1 Get Started with Android | 341 |
| 6.1.1 Install Android | 341 |
| 6.1.2 Calibrate Touch Screen | 342 |
| 6.1.3 Rotate Touch Screen | 343 |
| 6.1.4 Icons on the Status Bar | 344 |
| 6.1.5 Play MP3 | 344 |
| 6.1.6 Adjust Volumn | 345 |
| 6.1.7 Audio Recording | 345 |
| 6.1.8 SD WiFi | 346 |
| 6.1.9 USB Camera | 350 |
| 6.1.11 GPS | 351 |
| 6.1.12 Configure Ethernet | 352 |
| 6.1.13 3G | 357 |
| 6.1.14 USB Bluetooth | 362 |
| 6.1.14.1 Bluetooth Communication | 364 |
| 6.1.14.2 Transfer File to Cell Phone | 365 |
| 6.1.14.3 Transfer File to Mini6410-1405 | 370 |
| 6.1.15 USB Flash Drive | 372 |
| 6.1.16 Backlight Control | 374 |
| 6.1.17 Serial Port Assistant | 376 |
| 6.1.18 LED Test | 377 |
| 6.1.19 PWM Buzzer | 378 |
| 6.1.20 ADC Test | 379 |
| 6.1.21 I2C-EEPROM | 379 |
| 6.2 Set up Android Development Environment | 380 |
| 6.3 Set up Android Compiler | 381 |
| 6.3.1 Android Development and Compiler | 381 |



| | |
|--|-----|
| 6.3.2 Uncompress and Install Source Code | 381 |
| 6.4 Configure and Compile U-Boot..... | 383 |
| 6.5 Configure and Compile Linux Kernel | 384 |
| 6.6 Create Android | 384 |
| 6.7 Create or Run File System..... | 385 |
| 6.7.1 Make YAFFS2 Image..... | 386 |
| 6.7.2 Make UBIFS Image | 386 |
| 6.7.3 Make EXT3 Image..... | 387 |

Chapter 1 Introduction

This manual is intended to provide the user with an overview of the MINI6410 board, its benefits, features, specifications, and set up procedures.

1.1 Introduction to Mini6410-1405

The MINI6410-1405 development board is an excellent ARM11 board offering a comprehensive solution integrating both hardware and software. It is designed, developed and distributed by Friendly ARM. It applies the Samsung S3C6410 microprocessor and inherits all the features and benefits of our most popular MINI2440 products excelling in quality and easy to use with low cost. Compared to our previous products it has more reliable design and varied interfaces. These features make it easily and widely used in MID development, auto electronic devices, industrial applications, GPS systems and multimedia systems. It is good for educational training, embedded development and DIY as well.

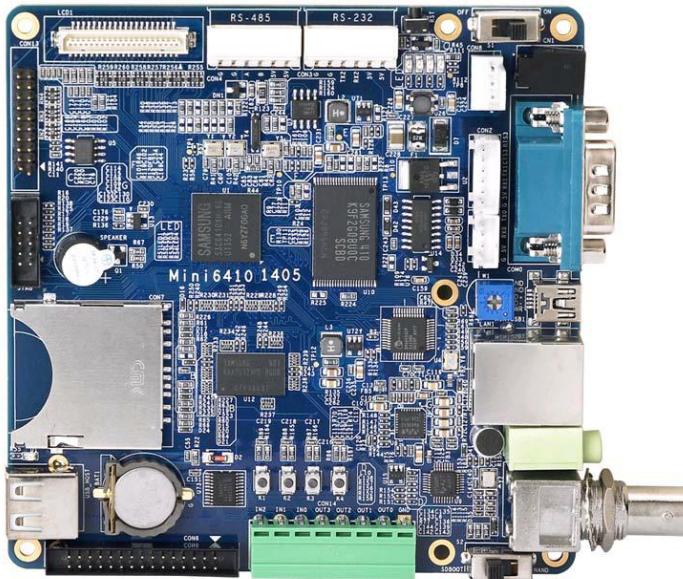
In addition we make the best use of the SD booting feature, by using our specially developed Superboot, enabling the board bootable from an SD card. With Superboot users can copy various systems (WindowsCE6/Linux/Android/uCos2) into an SD card (up to a maximum memory of 32G) and install them on the board without connecting to a PC, or run them on the SD card without burning systems onto the board.

Customers can get the latest information and news about our products by visiting:

<http://www.arm9.net>

1.1.1 Mini6410-1405 Overview

The Mini6410-1405 development board is a 110 x 110(mm) board equipped with a wide variety of connectors, interfaces and ports. Here's the layout:



Mini6410 – 1405

1.1.2 Mini6410-1405 Hardware Features

- **CPU**

- Samsung S3C6410A, ARM1176JZF-S, 533Mhz, maximum 667Mhz

- **DDR RAM**

- On board 256M DDR RAM (128M Optional)
- 32bit Data Bus

- **FLASH Memory**

- On board 256M/1GB SLC Nand Flash

- **LCD**

- Four-wire touch screen interface
 - Support black and white, 4 level grayscale, 16 level grayscale, 256-color、4096-color
- STN LCD, 3.5-inch to 12.1-inch, screen resolution 1024x768;
- Support black and white, 4 level grayscale, 16 level grayscale, 256-color, 64K-color,
 - true color TFT LCD, 3.5-inch to 12.1-inch, screen resolution 1024x768,

● Interfaces and External Accessories

- 1 x 100M Ethernet RJ-45 port (powered by the DM9000 network chip)
- 1 x DB9 RS232 and 1 x 2.54 mm spacing socket and two TTL sockets
- 1 x mini USB Slave 2.0 with OTG
- 1 x 3.5mm dual stereo audio output and single microphone input
- 1 x AV-IN
- 1 x USB Host 1.1 port
- 1 x standard SD card socket
- 5V power supply
- 1 x I2C-EEPROM chip (256 byte), for I2C Bus test
- 4 x USER LED (green)
- 4 x USER Button (interrupt pins, extended from board)
- 1 x Adjustable resistor, for AD conversion testing
- 1 x PWM buzzer
- Onboard real-time clock backup battery

- **Extended Interfaces**

- 1 x 10 pin 2.0mm spacing JTAG interface
- 2 x LCD Interfaces (one 41 pin connector compatible with MINI2440 LCD and the other 40 pin 2.0 mm spacing connector)
- 1 x 20pin 2.0mm spacing SDIO interface (can be connected to SD WiFi and include an SPI and an I2C)
- 1 x 30 pin 2.0mm spacing GPIO port
- 3 x IO input
- 4 x IO output

- **PCB Overview**

- PCB layer: 6
- 110 x 110 mm

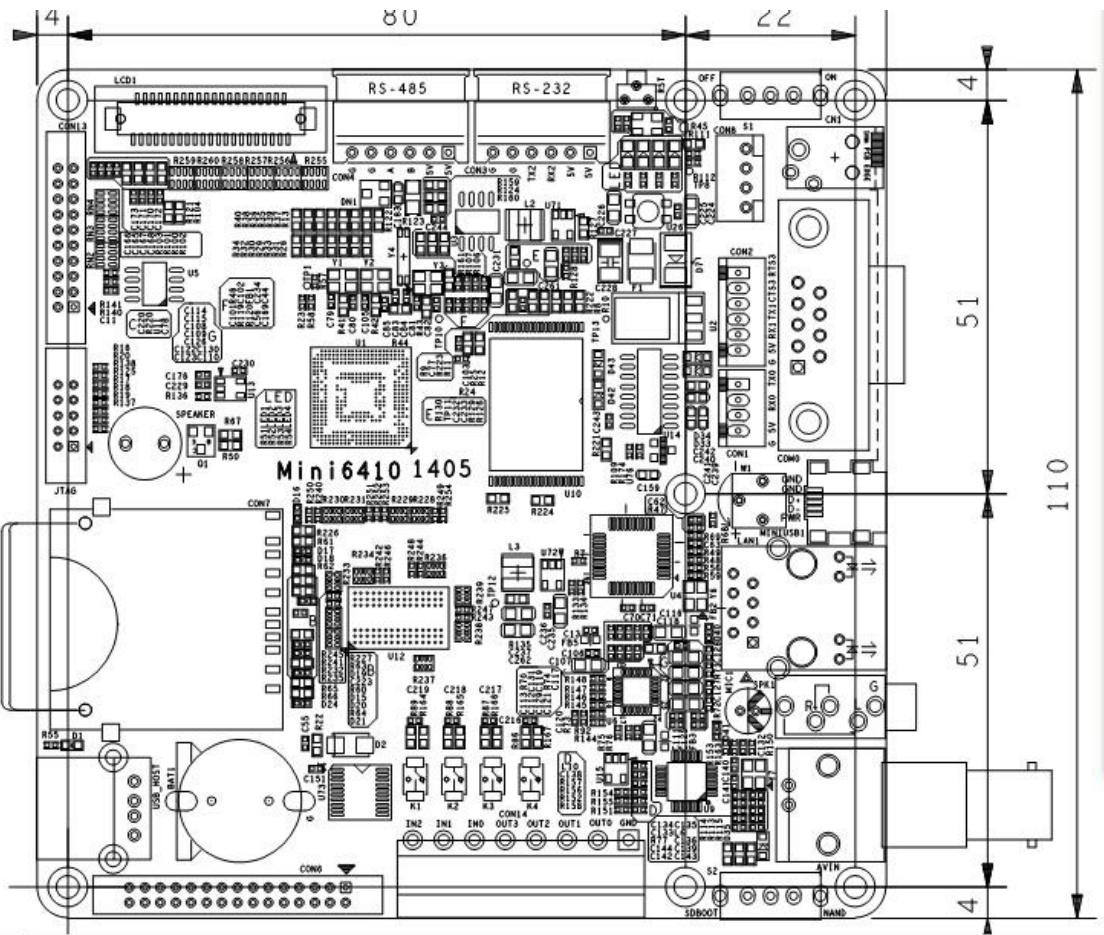
- **OS Support**

The Mini6410-1405 currently can work on the following operating systems:

- Linux2.6.38 + Qtopia-2.2.0 + QtE-4.7.0
- WindowsCE.NET 6.0(R3)
- Android 2.3

1.1.3 Board Dimension

Below is a detailed Mini6410-1405 board diagram:



1.1.4 Linux Features

- **Kernel Version**

- Linux 2.6.38

- **Boot Loader**

- U-boot-1.6.1: open source
- Superboot: developed by Friendly ARM, not open source

- **File Systems**

- YAFFS2: recommended

- UBIFS
- CRAMFS
- EXT2/3
- FAT32
- NFS

● Drivers

- Driver for 2 serial ports
- DM9000 driver
- Audio driver (WM9714)
- RTC driver
- Drivers for 4 User LEDs
- USB host driver
- LCD driver (it supports 3.5-inch, 4.3-inch, 7-inch, 8-inch, LCD2VGA1024x768, LCD2VGA800x600, LCD2VGA640x480 and EZVGA800x600)
- 4-wire touch screen driver, and 1-wire precise touching driver
- USB camera driver
- Drivers for USB mouse, keyboard, flash drive and portable hard disk
- SD card driver, up to 32 GB
- I2C-EEPROM driver
- ADC driver

- LCD backlight driver
- Watchdog driver (watchdog reset is cold reset)
- Multimedia drivers(including JPEG, FIMC, MFC, 2D/3D Accelerator, TVENC and TVSCALER)
- CMOS camera driver
- Backlight control, up to 127 levels. This allows users to adjust the board's backlight up to 127 levels and experience a gradually dim effect when turning it down.
- SPI driver

● Basic Applications and Utilities

- Busybox1.17 (Linux tool kit including basic Linux commands)
- Telnet, FTP and inetd (remote login tool)
- boa (web server)
- Madplay (command line mp3 player)
- Snapshot (command line screenshot tool)
- ifconfig, ping, route and so on (basic network commands)

● Graphic User Interface

- Qtopia-2.2.0: open source, including two versions: one for x86 and the other for ARM
- QtE-4.8.5: open source, only for ARM

● Qtopia Applications and Utilities

- A/D conversion test utility
- LED test utility
- User button test utility
- I2C-EEPROM read/write test utility
- LCD test utility
- Ping test utility
- USB camera preview and picture taking
- Audio recorder
- Web browser(konquor, open source)
- Watchdog test utility
- Network configuration utility
- Backlight control utility
- Language setting utility (English, Chinese and Japanese)
- Penpad utility (for touch pen testing)
- MMC/SD card and flash drive utility enabling auto mounting and unmounting
- Qt4 switcher
- Qtopia4 switcher
- SMPLAYER
- arm-none-linux -4.5.1-v6-vfp: cross compiler

1.1.5 WinCE 6.0 Features

● Version

- Windows CE Embedded 6.0

● Features

- Fast booting (boot up within 15 seconds)
- Support bootlogo download via USB, or SD Card only
- Configurable NBOOT header files: users can customize the process bar's color, position, length and width, and startup image's position and background.
- CMOS camera driver
- LED driver
- Driver for user buttons
- PWM buzzer driver
- LCD driver (it supports 3.5-inch, 4.3-inch, 7-inch, 8-inch, LCD2VGA1024x768, LCD2VGA800x600, LCD2VGA640x480 and EZVGA800x600)
- RTC driver
- DM9000 driver
- Large memory high speed SD/SDHC card driver(up to 32GB)
- Touch screen driver
- Audio input/output driver (WM9714 chip)
- Drivers for USB keyboard, mouse, flash drive

- Drivers for serial ports
- Multimedia driver (including JPEG, FIMC, 2D/3D Accelerator, MFC, TVENC and TVSCALER)
- USB WiFi
- USB Bluetooth
- 1-wire precise touch screen driver. It supports 4.3-inch to 21-inch touch screen
- Backlight control. This allows users to adjust the board's backlight up to 127 levels and experience a gradually dim effect when turning it down.

● Utility Highlights

- AV-IN: NTSC/PAL video input
- Serial port assistant
- User button test utility
- LED test utility
- PWM buzzer test utility
- Audio recorder
- OpenGL utility
- Autorun setup utility: to set up applications to be run at startup
- Painter: for touch screen test
- Notepad

1.1.6 Android Features

- **Linux Kernel**

- Linux-2.6.36

- **Boot Loader**

- U-boot-1.6

- **File Systems**

- FAT32
 - YAFFS2
 - UBIFS
 - EXT2/3

- **Android Version and Features**

- Android 2.3.4
 - 2D/3D Accelerator
 - SD WiFi, USB WiFi
 - GPS
 - Support USB 3G Card: more than one hundred kinds of USB 3G cards which cover all three systems: WCDMA, CDMA2000 and TD-SCDMA etc.
 - Flash drive plug and play: up to a maximum memory of 32G
 - USB Bluetooth
 - 1-wire precise touch screen

- Backlight control. This allows users to adjust the board's backlight up to 127 levels and experience a gradually dim effect when turning it down
- GUI utility for Ethernet configuration, enabling both auto and manual IP setup
- AV-IN

1.2 Board Schematics

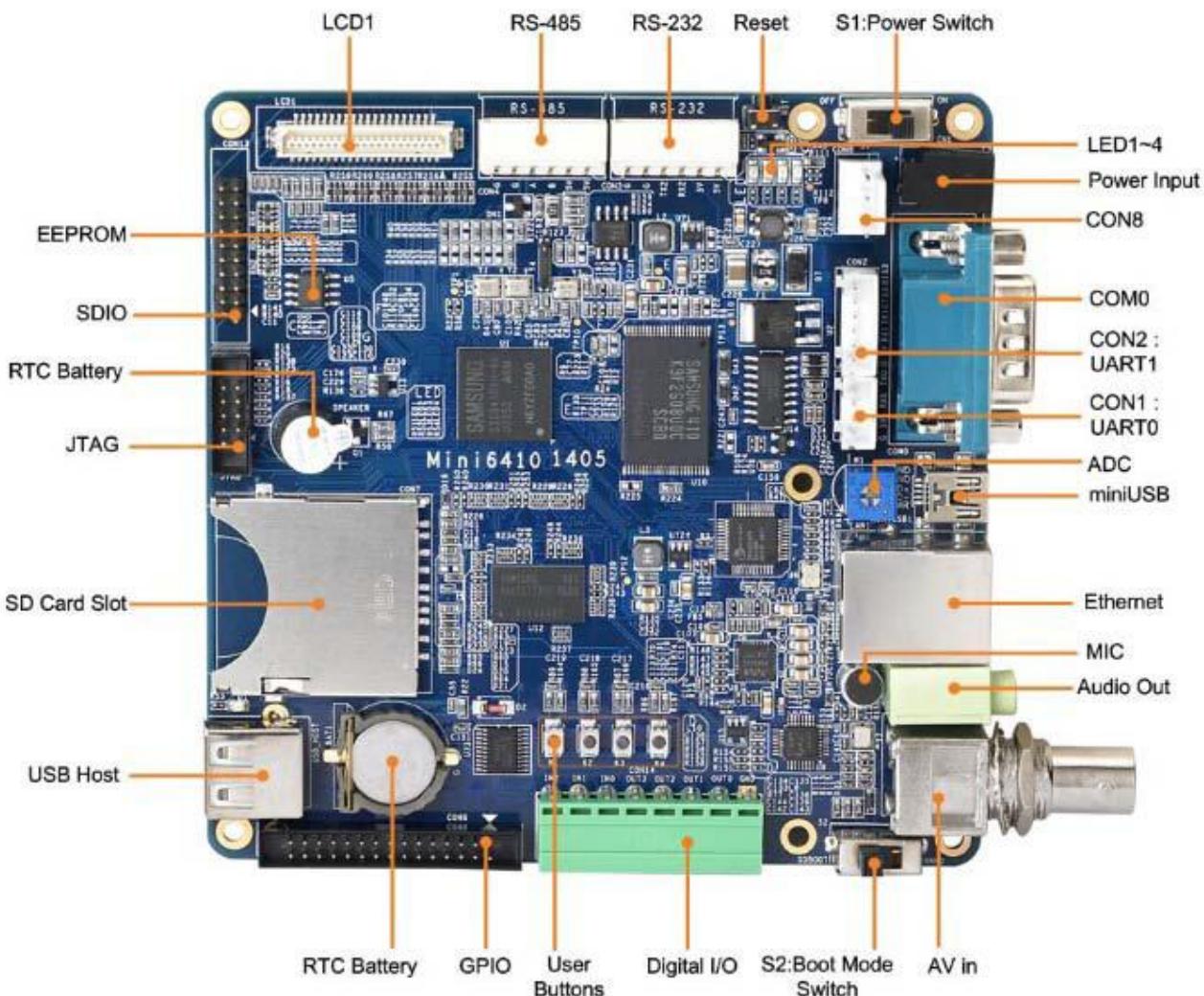
1.2.1 Jumper

There are no jumpers on this board. This design makes the board easy for users.

1.2.2 Schematic

The Mini6410-1405 schematic is presented as below. It includes most of the popular interfaces, ports, IOs and buses on a 110 x 110 mm board.

Note: in the diagram below those orange points indicate the first pins of those interfaces/ports/connectors accordingly.



1.3 Interface Specifications

This section describes in detail each interface/port on the board. For more details users can refer to the complete schematics (in PDF and Protel99SE) in the CDs shipped together with this product.

1.3.1 Address Space

The following data is from the S3C6410 data sheet 2.2.

| Address | | Size (MB) | Description |
|-------------|-------------|-----------|---------------------|
| 0x0000_0000 | 0x07FF_FFFF | 128MB | BOOT |
| 0x0800_0000 | 0x0BFF_FFFF | 64MB | ROM |
| 0x0C00_0000 | 0x0FFF_FFFF | 128MB | Stepping Stone(8KB) |
| 0x1000_0000 | 0x17FF_FFFF | 128MB | |
| 0x1800_0000 | 0x1FFF_FFFF | 128MB | DM9000AEP |
| 0x2000_0000 | 0x27FF_FFFF | 128MB | |
| 0x2800_0000 | 0x2FFF_FFFF | 128MB | |
| 0x3000_0000 | 0x37FF_FFFF | 128MB | |
| 0x3800_0000 | 0x3FFF_FFFF | 128MB | |
| 0x4000_0000 | 0x47FF_FFFF | 128MB | |
| 0x4800_0000 | 0x4FFF_FFFF | 128MB | |
| 0x5000_0000 | 0x5FFF_FFFF | 256MB | 128M DDR RAM |
| 0x6000_0000 | 0x6FFF_FFFF | 256MB | |

1.3.2 Power

The Mini6410-1405 is powered by an external 5V power supply. It has two power inlets: CN1 is for 5V power adapter and the white CON8 is a 4 pin socket used to connect an external power supply when the board is embedded in a closed box.

| CON8 | PIN Spec |
|------|----------|
| 1 | VDD5V |
| 2 | GND |
| 3 | GND |
| 4 | VDDIN |

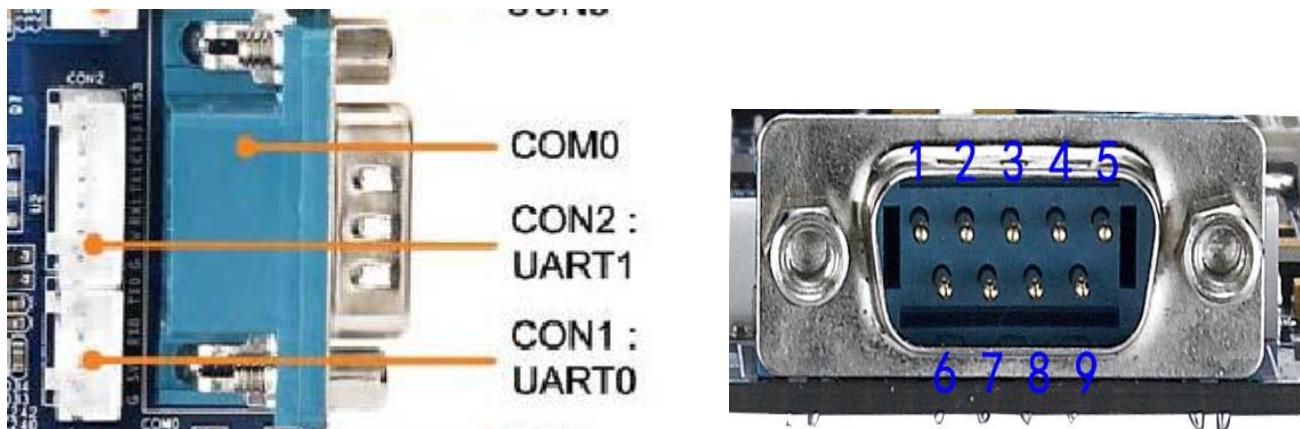
Note: when connected to an extended cable, the S1 switch works too.

1.3.3 Serial Port

S3C6410 has 4 serial ports: UART0, 1, 2 and 3. UART1 is a 5-wire serial port and the other two are 3-wire serial ports.

On this board, UART0 and UART2 are converted to RS232 levels to COM0 and CON3

respectively.



| CON1,3,4 | PIN Spec (TTL) | CON2 | PIN Spec (TTL) | COM0 | PIN Spec (RS232) |
|----------|----------------|------|----------------|------|------------------|
| 1 | TXD | 1 | RTSn | 1 | NC |
| 2 | RXD | 2 | CTSn | 2 | RSRXD |
| 3 | 5V | 3 | TXD | 3 | RSTXD |
| 4 | GND | 4 | RXD | 4 | NC |
| | | 5 | 5V | 5 | GND |
| | | | GND | 6 | NC |
| | | | | 7 | RSCTSs |
| | | | | 8 | RSRTSs |
| | | | | 9 | NC |

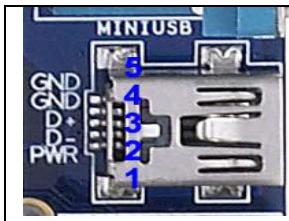
Note: NC means floating

1.3.4 USB Interface

The Mini6410-1405 board has two kinds of USB interface. One is a USB host 1.1 which is the same as a USB interface on a PC and can connect to a USB camera, keyboard, mouse, flash drive and other USB devices. The other is an OTG mini USB 2.0 which is usually used to download programs to a target board. When the board runs WinCE it can synchronize with a Windows via ActiveSync. For Linux there are no programs for synchronization for

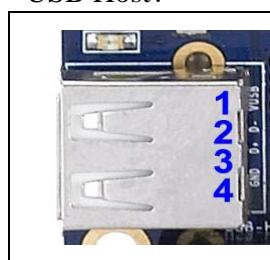
now.

Mini USB interface:



| Mini USB | PIN Spec |
|----------|----------|
| 5 | GND |
| 4 | OTGID |
| 3 | D+ |
| 2 | D- |
| 1 | Vbus |

USB Host:



| USB Host | PIN Spec |
|----------|----------|
| 1 | 5V |
| 2 | D- |
| 3 | D+ |
| 4 | GND |

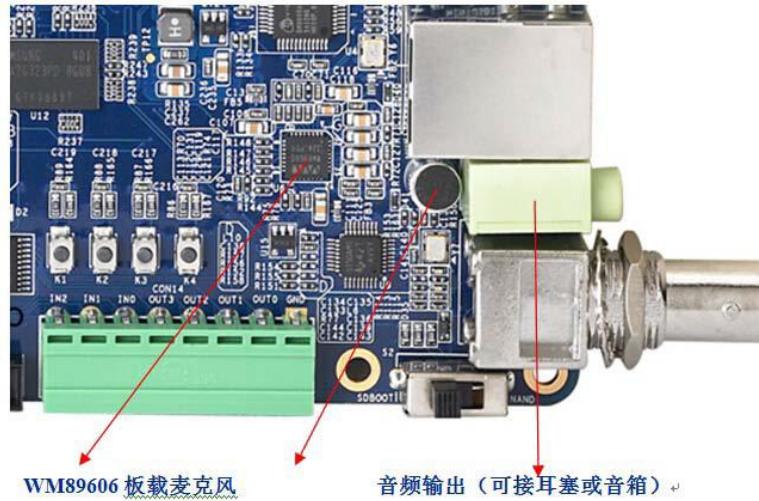
1.3.5 Network Interface

The Mini6410-1405 incorporates a DM9000 chip and can communicate with 10/100M networks. The RJ45 connector includes coupling filters and does not need transformers. With a common network cable, you can connect a router or switch to the Mini6410-1405.

1.3.6 Audio Interface

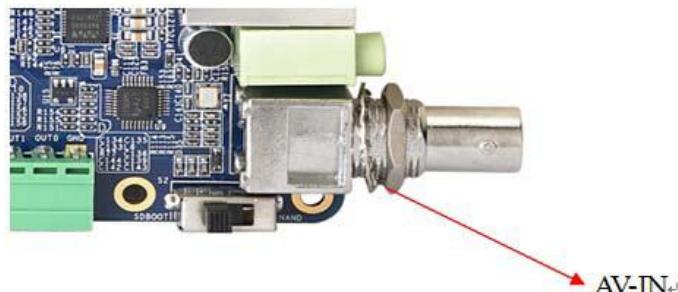
The S3C6410 supports I2S/PCM/AC97. The board has an AC97 interface which uses WM8960 as the CODEC chip.

The audio output is a 3.5 mm spaced green plug and the input is an on-board microphone. To get better audio quality please move the microphone as close as possible to the audio source when recording.



1.3.7 AV-IN

The Mini6410-1405 has an AV-IN BNC port which supports NTSC/PAL signals.

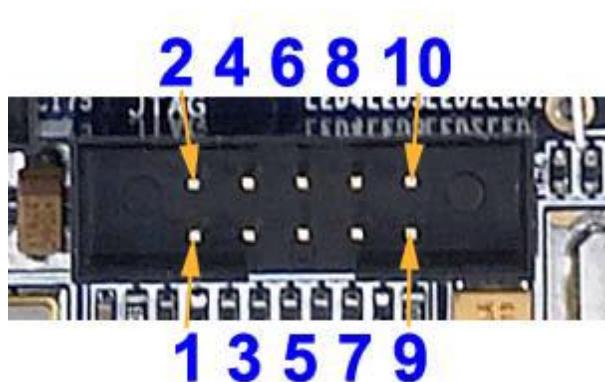


1.3.8 JTAG Interface

When a board just comes off from production lines it is just a bare board without any data and we usually have to burn the first program to it through the JTAG interface. However since the S3C6410 supports booting from SD card the JTAG is not significant to users any more. Now the JTAG is more often used for debugging. In fact, most of the widely used utilities in markets like JLINK, ULINK and other simulators actually work via the JTAG interface. A

standard JTAG has 4 signals :TMS, TCK, TDI and TDO which are test mode select input, test clock, test data input and test data output. These 4 signal lines plus a power line and a ground line form 6 lines in total. In order for testing, most simulators even have a reset signal. Therefore, a standard JTAG is meant to have those signal lines, and it does not mean whether it is 20Pin or 10Pin. As long as a JTAG interface has those signal lines, it will be a standard JTAG interface. The Mini6410-1405 has a 10Pin JTAG interface which has complete standard JTAG signals.

Notes: for beginners who just want to focus on Linux or WinCE development, the JTAG interface has no significance because most development boards already have a complete BSP which includes commonly needed serial ports, network port and USB port. When a board runs with Linux or WinCE installed, users can fully utilize more convenient functions and utilities provided by the operating system to debug. They do not need a JTAG. Even if you can trace your programs it will be extremely tough to step debug because it will go into the operating system. This is not an easy job.



| JTAG Spec | | | | |
|-----------|---|---|---|----|
| 2 | 4 | 6 | 8 | 10 |



| | | | | |
|------|--------|-----|-----|-----|
| 3.3V | nRESET | TDO | GND | GND |
| 1 | 3 | 5 | 7 | 9 |
| 3.3V | nTRST | TDI | TMS | TCK |

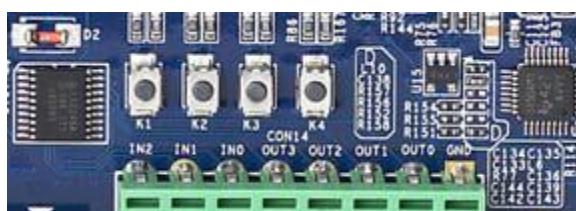
1.3.9 LED

A LED is commonly used as a status indicator. The Mini6410-1405 has 4 user programmable LEDs which are directly connected to GPIO. The LEDs will be on at a low level voltage. Detailed information is as follows

| LED4...1 | GPIO | LED4 | LED3 | LED2 | LED1 |
|----------|------|------|------|------|------|
| | GPK7 | GPK6 | GPK5 | GPK4 | |

1.3.10 User Key

The Mini6410-1405 has 4 user keys. All of them are directly connected to CPU interrupt pins and will be triggered at a low level voltage. They can also be multiplexed to GPIO and other function interfaces. To multiplex them users can extend them through CON12. These 8 keys and CON12 are defined as follows



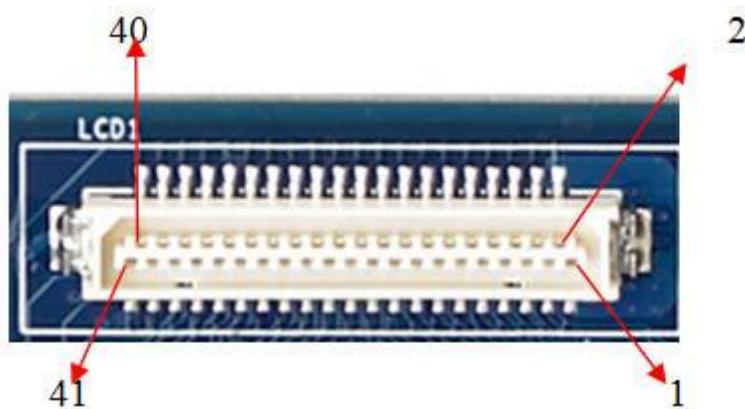
| CON12 | 1 | 2 | 3 | 4 |
|---|-------|-------|-------|-------|
| Key | K1 | K2 | K4 | K4 |
| Interrupt | EINT0 | EINT1 | EINT2 | EINT3 |
| Multiplexed GPIO | GPN0 | GPN1 | GPN2 | GPN3 |
| Note: CON12.9 is 3.3V and CON12.10 is GND | | | | |

1.3.11 LCD Interface

The Mini6410-1405 has a 0.5mm spacing 41 pin connector.

The LCD connector has most of the commonly used control signals (line scan, clock, enable/disable) and complete RGB data signals (RGB output is 8:8:8 and can support LCDs up to 1.6M pixels). It has a PWM output and a reset signal (nRESET). LCD_PWR is the backlight switch signal.

37, 38, 39 and 40 are a 4 wire touch screen interface which can be directly connected to a 4 wire resistor touch screen.



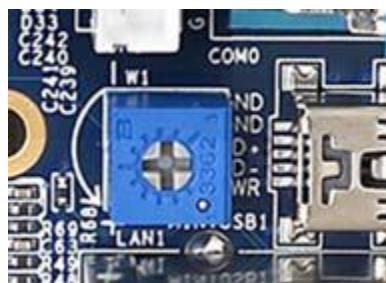
| LCD | PIN Spec | LCD2 | PIN Spec |
|-----|----------|------|----------|
| 1 | 5V | 2 | 5V |
| 3 | VD0 | 4 | VD1 |
| 5 | VD2 | 6 | VD3 |
| 7 | VD4 | 8 | VD5 |
| 9 | VD6 | 10 | VD7 |
| 11 | GND | 12 | VD8 |
| 13 | VD9 | 14 | VD10 |
| 15 | VD11 | 16 | VD12 |
| 17 | VD13 | 18 | VD14 |
| 19 | VD15 | 20 | GND |

| | | | |
|----|------------|----|--------------|
| 21 | VD16 | 22 | VD17 |
| 23 | VD18 | 24 | VD19 |
| 25 | VD20 | 26 | VD21 |
| 27 | VD22 | 28 | VD23 |
| 29 | GND | 30 | GPE0/LCD_PWR |
| 31 | PWM1/GPF15 | 32 | nRESET |
| 33 | VDEN/VM | 34 | VSYNC |
| 35 | HSYNC | 36 | VCLK |
| 37 | TSXM | 38 | TSXP |
| 39 | TSYM | 40 | TSYP |
| | | 41 | GND |

1.3.12 ADC Input

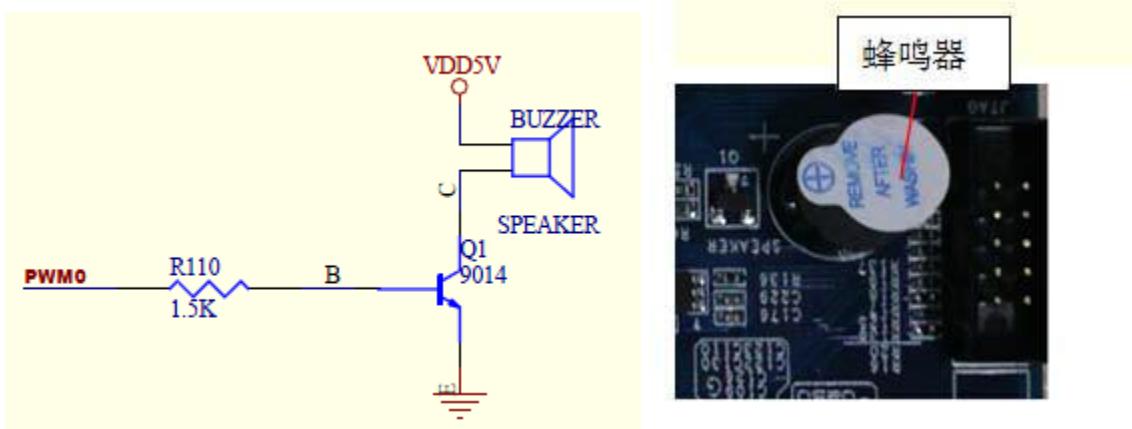
The Mini6410-1405 has 4 A/D conversion channels. AIN0 is connected to the adjustable resistor W1; AIN1, 2 and 3 are extended via CON6's 27, 28 and 29. The S3C6410's AD conversion can be configured to 10-bit/12-bit.

W1 is placed close to one edge of the board and will not be hidden even when the board is equipped with a 4.3-inch LCD.



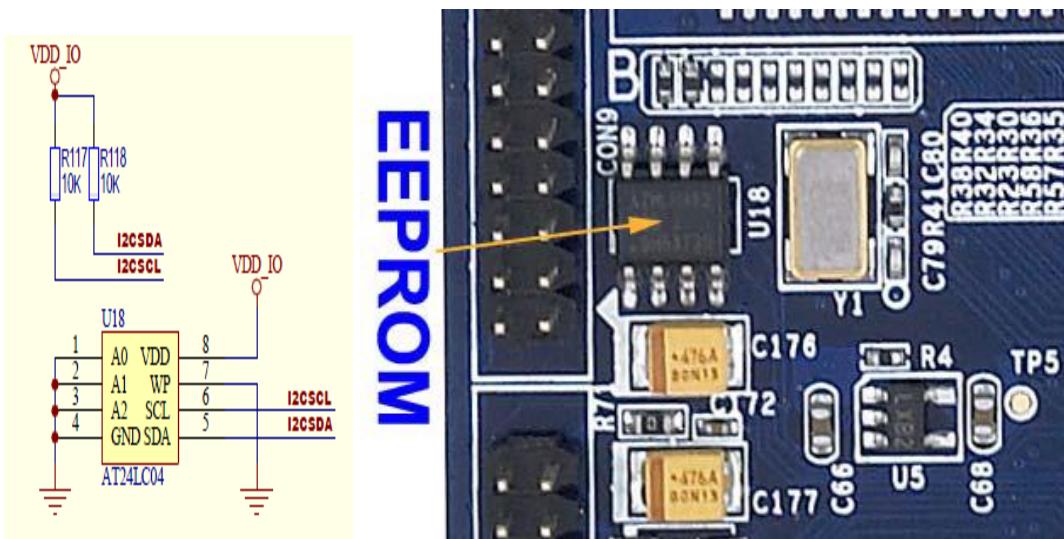
1.3.13 PWM Buzzer

The on-board SPEAKER is controlled by PWM, the diagram is shown below. PWM0 corresponds to GPF14 which can be configured as PWM output via software or used as a GPIO.



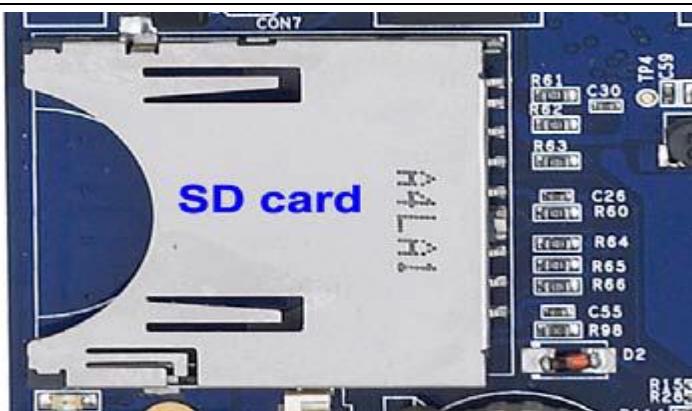
1.3.14 I2C-EEPROM

The Mini6410-1405 has an EEPROM AT24C08 connected to CPU's I2C. It has 256 bytes memory and is mainly for testing I2C bus.



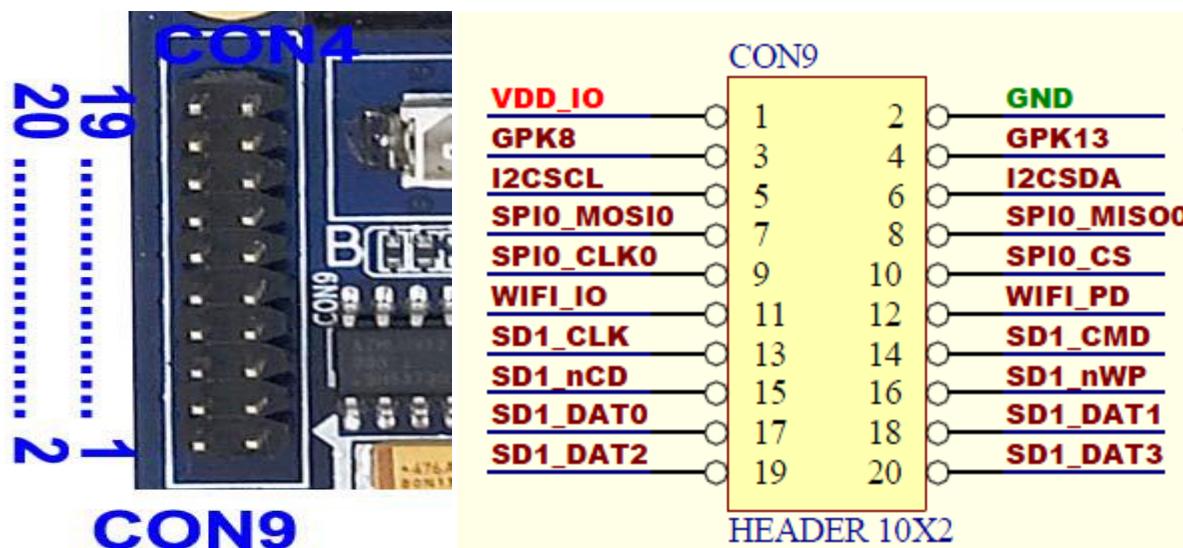
1.3.15 SD Card

The S3C6410 has two SDIO interfaces. SDIO0 is commonly used for SD cards. The Mini6410-1405 makes a standard SD card slot by extending SDIO0. It supports SDHC i.e. high speed large memory SD cards.



1.3.16 SDIO-II/SD-WiFi

The other SDIO of the S3C6410 is extended via CON9. It is a 2.0 mm spacing 20 pin connector and includes an SPI, an I2C and 4 GPIOs.

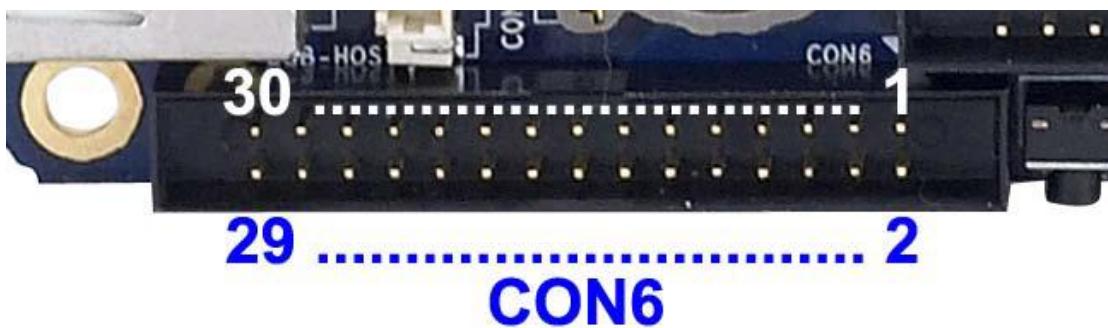


| CON9 | PIN Spec | CON9 | PIN Spec |
|------|---------------|------|---------------|
| 1 | VDD/3.3V | 2 | GND |
| 3 | GPK8 | 4 | GPK13 |
| 5 | I2CSCL | 6 | I2CSDA |
| 7 | SPI0_MOSI0 | 8 | SPI0_MISO0 |
| 9 | SPI0_CLK0 | 10 | SPI0_CS |
| 11 | GPP10/WiFi_IO | 12 | GPP11/WiFi_PD |
| 13 | SD1_CLK | 14 | SD1_CMD |
| 15 | SD1_nCD | 16 | SD1_nWP |
| 17 | SD1_DAT0 | 18 | SD1_DAT1 |
| 19 | SD1_DAT2 | 20 | SD1_DAT3 |

| | | | |
|----|----------|----|----------|
| 17 | SD1_DAT0 | 18 | SD1_DAT1 |
| 19 | SD1_DAT2 | 20 | SD1_DAT3 |

1.3.17 GPIO

GPIO is the abbreviated form of General Purpose Input Output. The Mini6410-1405 has a 30 Pin 2.0mm spacing GPIO interface, i.e. CON6. In fact, CON6 has not only quite a few GPIO pins but also some CPU pins such as AD input, DAC and so on. The SPI interface, I2C interface, interrupts and some others are all GPIOs, but they are marked as special function interfaces. They can be configured for other purposes too by setting related CPU registers.



| CON6 | PIN Spec | Description | CON6 | PIN Spec | Description |
|------|----------|--------------------------------|------|-----------|--------------------------------|
| 1 | 3.3V | Power | 2 | GND | Ground |
| 3 | GPE1 | Available, can be used as GPIO | 4 | GPE2 | Available, can be used as GPIO |
| 5 | GPE3 | Available, can be used as GPIO | 6 | GPE4 | Available, can be used as GPIO |
| 7 | GPM0 | Available, can be used as GPIO | 8 | GPM1 | Available, can be used as GPIO |
| 9 | GPM2 | Available, can be used as GPIO | 10 | GPM3 | Available, can be used as GPIO |
| 11 | GPM4 | Available, can be used as GPIO | 12 | GPM5 | Available, can be used as GPIO |
| 13 | GPQ1 | Available, can be used as GPIO | 14 | GPQ2 | Available, can be used as GPIO |
| 15 | GPQ3 | Available, can be used as GPIO | 16 | GPQ4 | Available, can be used as GPIO |
| 17 | GPQ5 | Available, can be used as GPIO | 18 | GPQ6 | Available, can be used as GPIO |
| 19 | SPI1_CLK | Available, can be used as GPIO | 20 | SPI1_MISO | Available, can be used as GPIO |
| 21 | SPI1_CS | Available, can be used as GPIO | 22 | SPI1_MOSI | Available, can be used as GPIO |
| 23 | EINT6 | Available, can be used as GPIO | 24 | EINT9 | Available, can be used as GPIO |
| 25 | EINT11 | Available, can be used as GPIO | 26 | EINT16 | Available, can be used as GPIO |

| | | | | | |
|---|------|-------------------------|----|---------|-------------------------|
| 27 | AIN1 | Analog input1, 0-0.3.3V | 28 | AIN2 | Analog input2, 0-0.3.3V |
| 29 | AIN3 | Analog input3, 0-0.3.3V | 30 | DACOUT1 | |
| Note: DACOUT1 is TV output. It needs to be connected to a magnifier for TV output | | | | | |

Chapter 2 Get Started

This chapter will guide you to experience our Mini6410-1405 step by step. We strongly recommend our users to read them carefully. Although those details may seem tedious and boring they list very useful references and notes which will be of great help for developers.

By default, all our systems have been preinstalled Linux (located in the shipped CDs' directory /images/Linux are **u-boot**, **zImage**, **root-qtopia-qt4.img** etc) therefore you could easily play once you get it.

2.1 System Setup and Configuration

2.1.1. Boot Options

Switch “S2” to the “SDBOOT” side, the system will boot from SD/SDHC card.

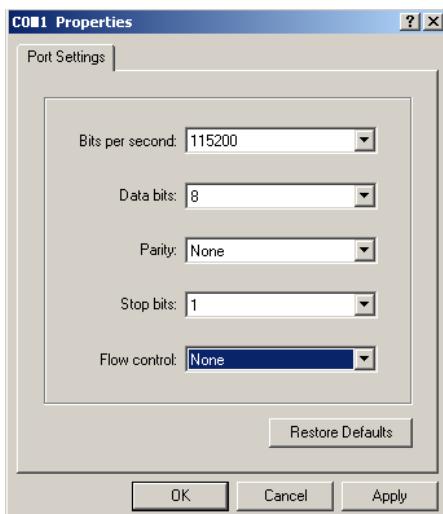
Switch “S2” to the “NAND” side, the system will boot from Nand Flash.

2.1.2 Connect Peripherals

- Connect the Mini6410-1405 board's serial port0 to a PC's serial port with the shipped crossover serial cable (**blue one**) in the package
- Connect the shipped 5V power supply
- Connect an LCD touch panel (if you have)

2.1.3 Set Up Hyper Terminal

Setup your serial terminal like this:



2.2 Burn BIOS to SD Card

Note: common SD cards are formatted to FAT32 by default. If there is a huge amount of data in a SD card, auto burning will damage the data. To avoid this issue, in Vista/Windows7 we automatically format a SD card to two sections: one is FAT32 (named “FriendlyARM”) for users' data and the other (by default 130M) left for the bootloader.

In fact, Vista/Windows7's system security policies don't permit unauthorized users to

start auto burning an SD card thus ordinary users need to format the SD card first and then burn data into it. For WindowsXP users it is reported that quite a lot of them don't want to upgrade their systems to Windows7 nor do they care much about this issue therefore we just set the burning mode to auto burning, the same as what Samsung does.

2.2.1 Introduction to BIOS

● U-Boot

Samsung offers a U-Boot that can download files via USB. We improved this function and made it open sourced for all embedded development lovers and fans. The main features are as follows:

1. Add a pull-down menu, which is similar to the one in Superboot's USB download menu.
2. Add configuration options for SD card booting
3. Support direct downloading and burning of the YAFFS2 file system
4. Support burning of NBOOT for WindowsCE Boot Loader
5. Support burning of WindowsCE image
6. Support burning of independent applications
7. Support shell command line

Note: the open source u-boot can only support SLC nand flash, doesn't support MLC nand flash.

● Superboot

Besides the U-Boot, we developed another powerful bootloader – “Superboot”. It is not open source. Below is a table list of features of different bootloaders

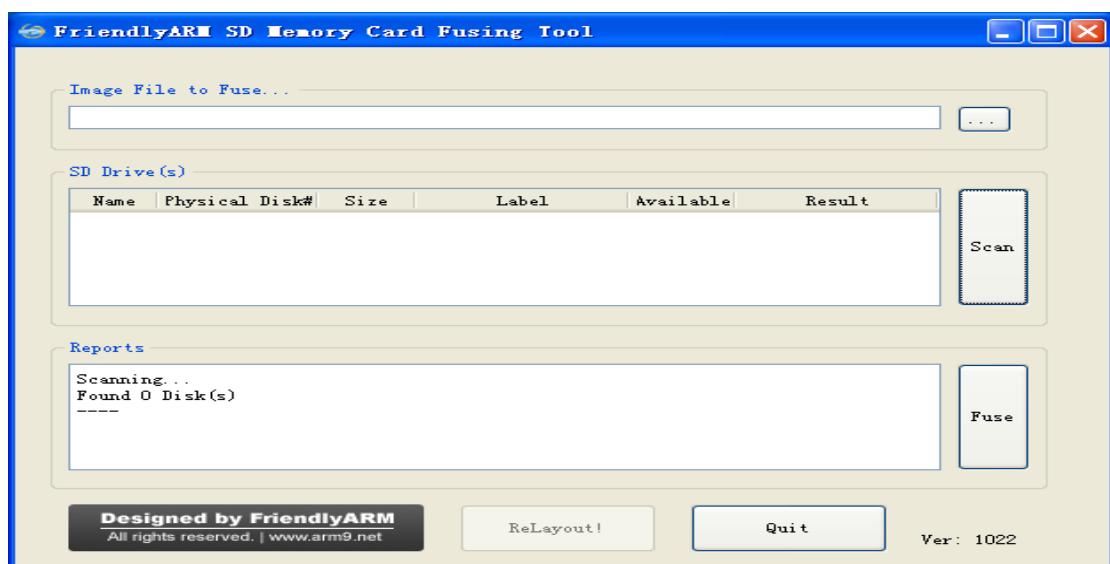
| Items | Superboot | U-Boot by Friendly ARM | Other U-Boot |
|---|--------------------------------|------------------------|--------------|
| Developer | Friendly ARM | Friendly ARM | Others |
| Download Menu | Yes | Yes | No |
| Auto detection of 128M/256M DDR RAM | Yes | No | No |
| Large size image file (> RAM) | Yes | No | No |
| USB download StepLoader, such as Nboot | Yes | Yes | Yes |
| USB download Linux kernel | Yes | Yes | Yes |
| USB download Yaffs2 | Yes | Yes | No |
| USB download UBIFS | Yes | No | No |
| USB download EBOOT.nb0 | No, unnecessary | No, unnecessary | Yes |
| USB download WindowsCE image NK.bin | Yes | No | No |
| USB download WindowsCE image NK.nb0 | No, we think it is unnecessary | Yes | No |
| USB download WindowsCE boot logo | Yes | No | No |
| USB download standalone programs | Yes | Yes | Yes |
| Boot Configuration | Yes | Yes | Yes |
| SD card (< 2G) | Yes | Yes | Yes |
| SDHC | Yes | No | No |
| SD card quick-auto-burning bootloader | Yes | No | No |
| SD card quick-auto-burning Linux kernel | Yes | No | No |
| SD card quick-auto-burning Yaffs2 | Yes | No | No |
| SD card quick-auto-burning UBIFS | Yes | No | No |
| SD card quick-auto-burning WindowsCE image NK.bin | Yes | No | No |
| SD card quick-auto-burning WindowsCE boot logo | Yes | No | No |
| SD card quick-auto-burning standalone programs | Yes | No | No |
| Running Linux on SD card | Yes | No | No |
| Running WindowsCE on SD card | Yes | No | No |
| Running Android on SD Card | Yes | No | No |
| Running standalone programs on SD card | Yes | No | No |
| SD card rapid reading and burning (see note1) | Yes | No | No |

2.2.2 Burning BIOS in WindowsXP

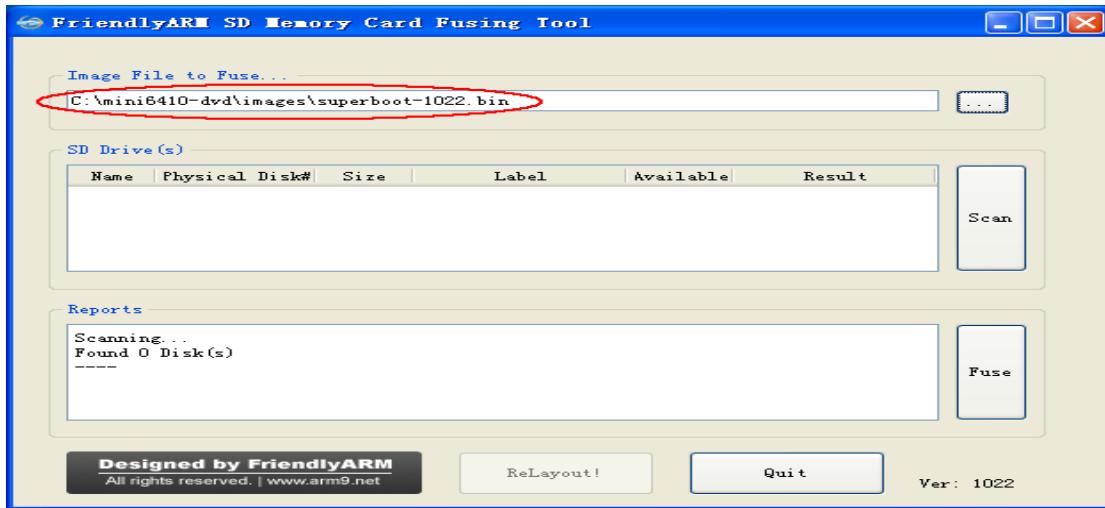
Note: users complained that some notebook's integrated SD card reader cannot work properly with card burning or reading. So far we haven't encountered this issue and we suggest that you should try an external usb card reader in this case.

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

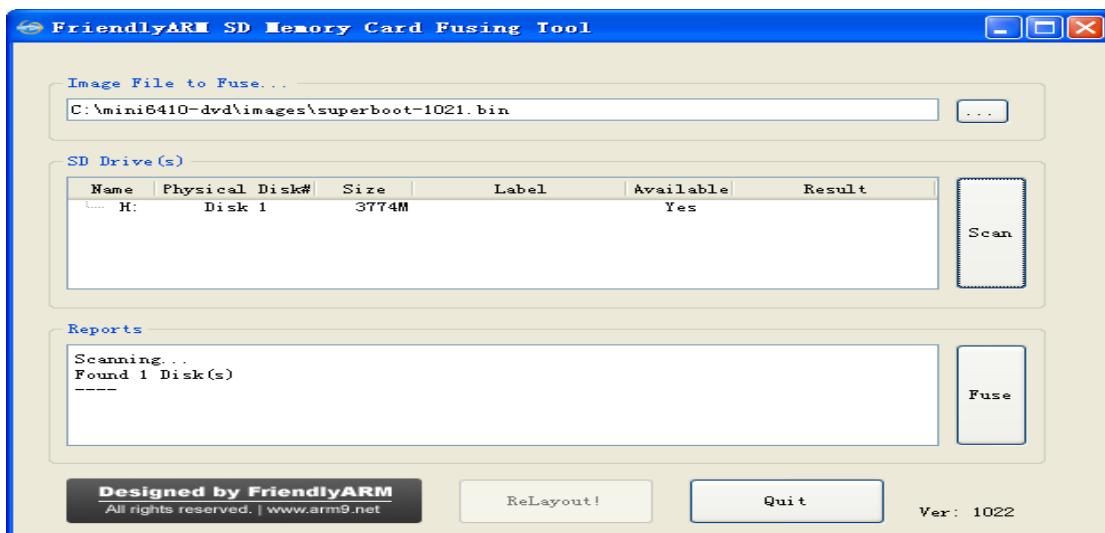
Step1: launch the SD-Flasher.exe (under “\tools\”). Note: the “ReLayout” button is disabled. We set it purposely in WindowsXP.



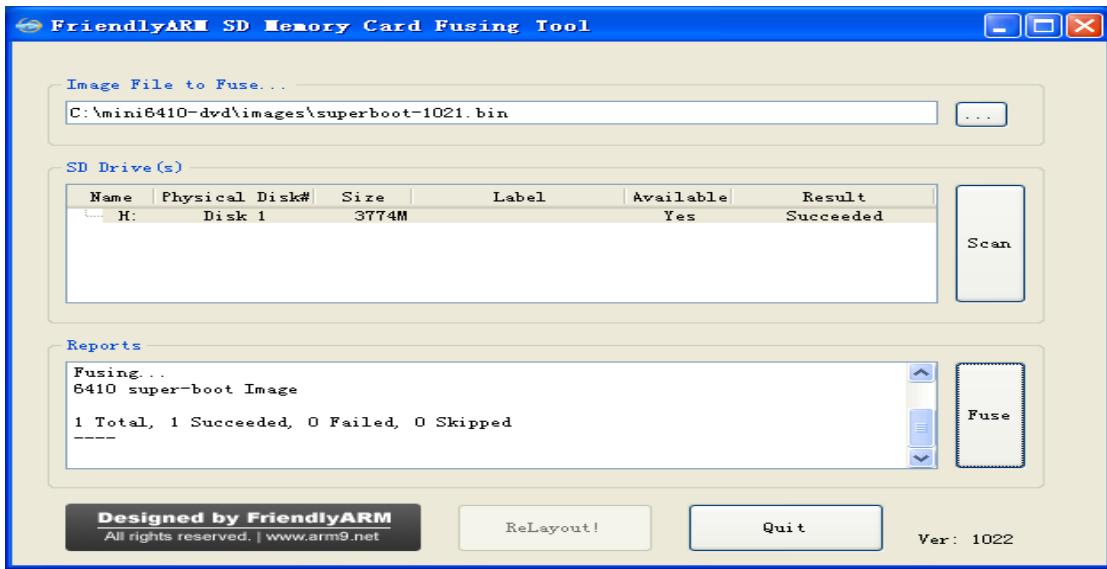
Step2: click on  to select Superboot



Step 3: insert a FAT32 SD card into your SD card socket (you can also use a USB card reader to connect to a PC) and click on “Scan”, all recognized SD cards will be listed.



Step 4: click on “Fuse”, Superboot will be burned into your SD card



The Superboot in your SD card is invisible. To verify it you can insert your SD card into your board's SD card socket and switch S2 to the "SDBOOT" mode, reboot your board and if LED1 is flashing it is indicating that your Superboot is functioning.

Note: if you use a TF card it might not work properly due to bad connection, so we suggest a common SD card be used.

2.2.3 Burn BIOS in Windows 7

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

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

Step1: launch the SD-Flasher.exe (under "\tools\"). Note: you need to open it as

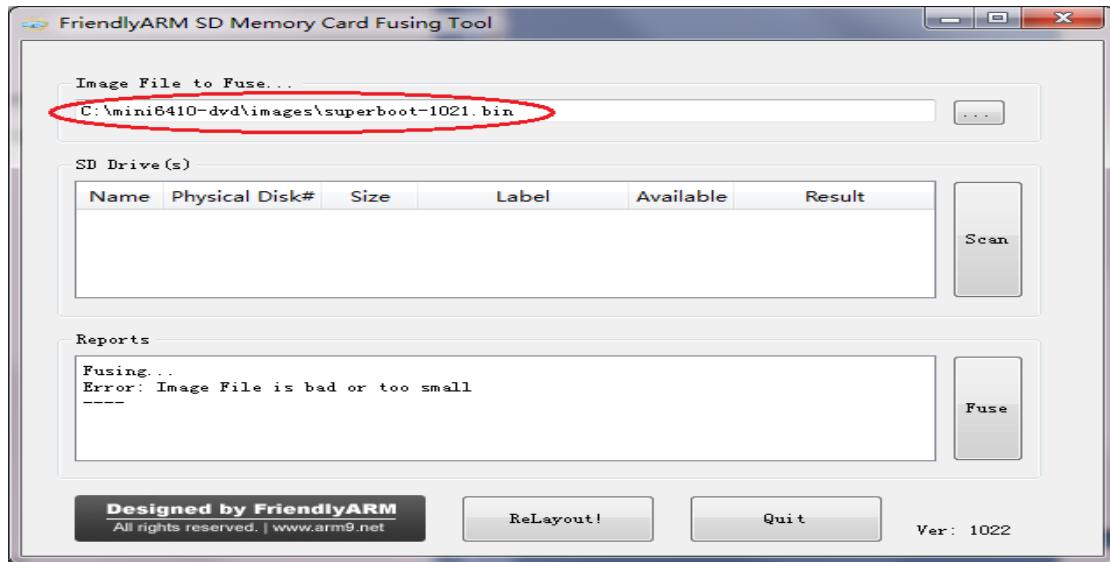
“Administrator”



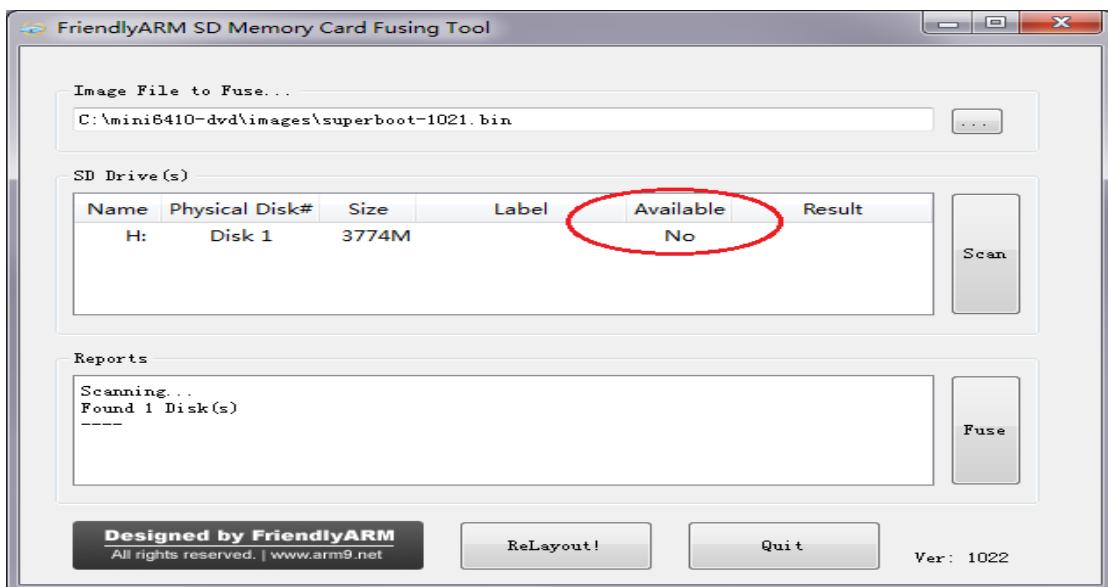
Below is the dialog you will see after it is started. Note: the “ReLayout” is enabled and we will format the SD card with this function.



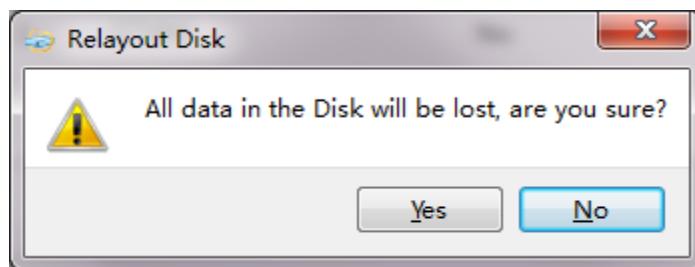
Step2: click on  to select Superboot



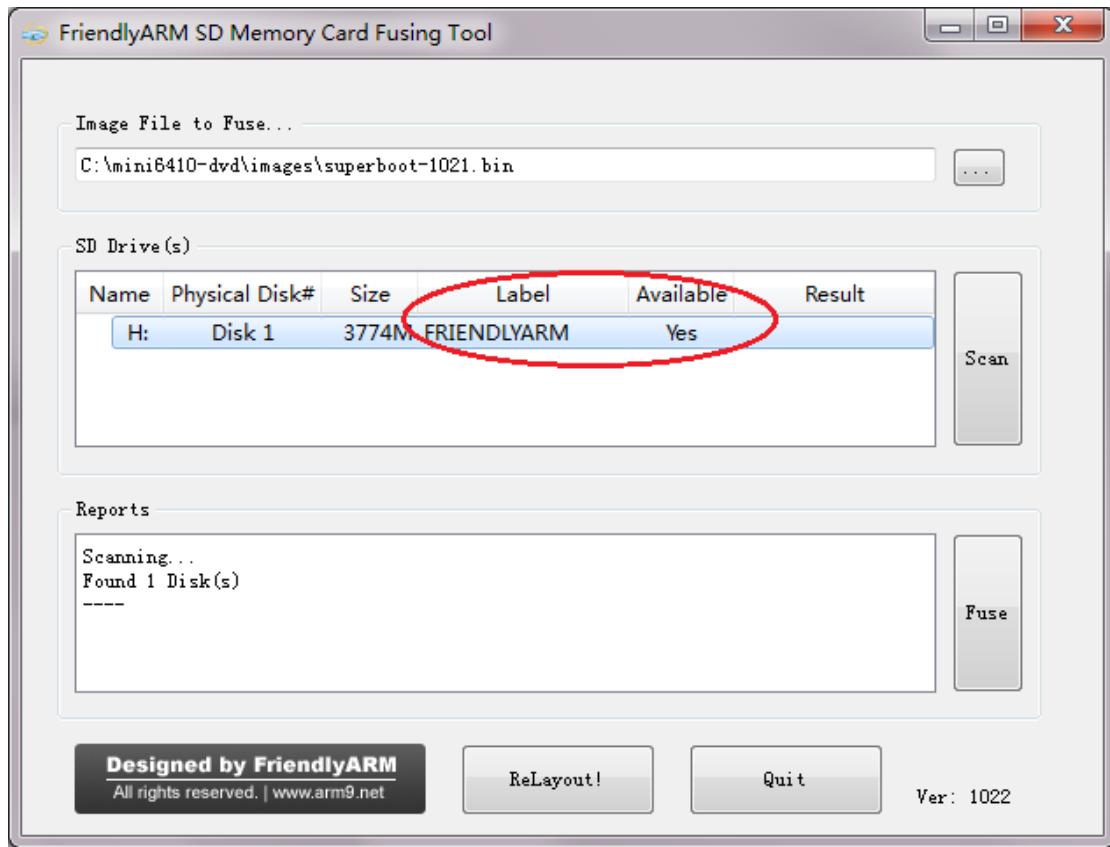
Step3: insert a FAT32 SD card into your SD card socket (you can also use a USB card reader to connect to a PC), backup your data in the card and click on “Scan”, all recognized SD cards will be listed. For now, the SD card cannot be burned (circled by red)



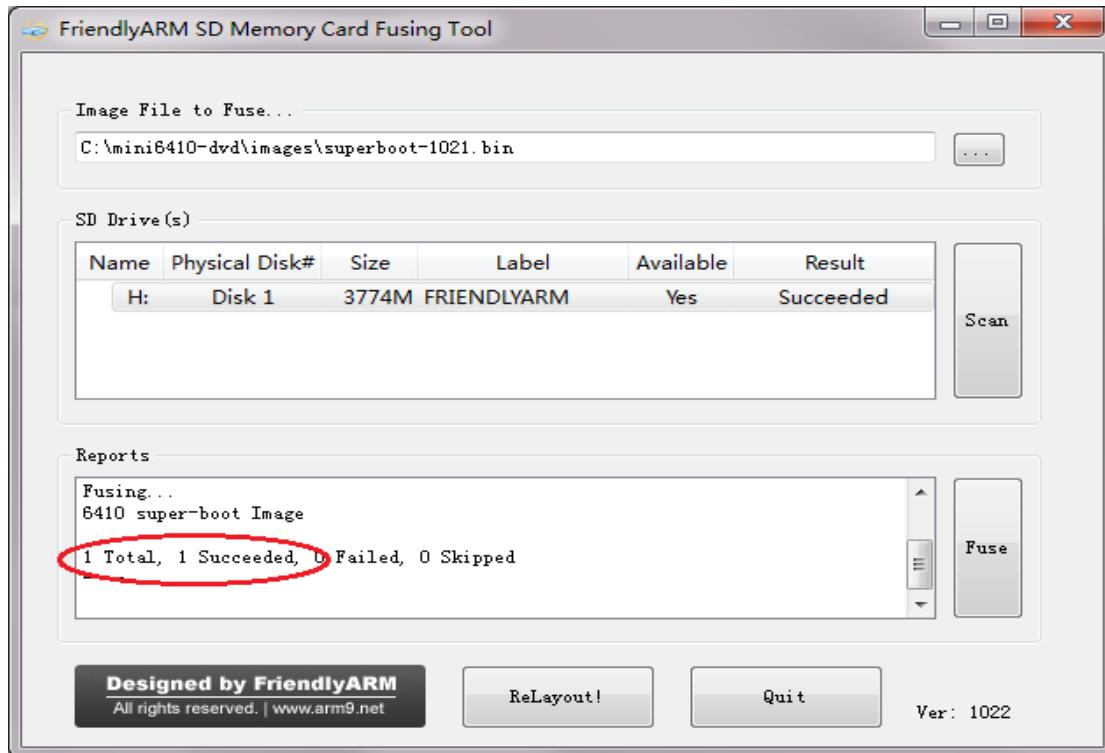
Step4: click on “ReLayout”, the following dialog will pop up prompting you that the data in your card will be lost. Just click on “Yes”



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



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



The Superboot in your SD card is invisible. To verify it you can insert your SD card into your board's SD card socket and switch S2 to the "SDBOOT" mode, reboot your board and if LED1 is flashing it is indicating that your Superboot is functioning.

2.3 Play with Superboot

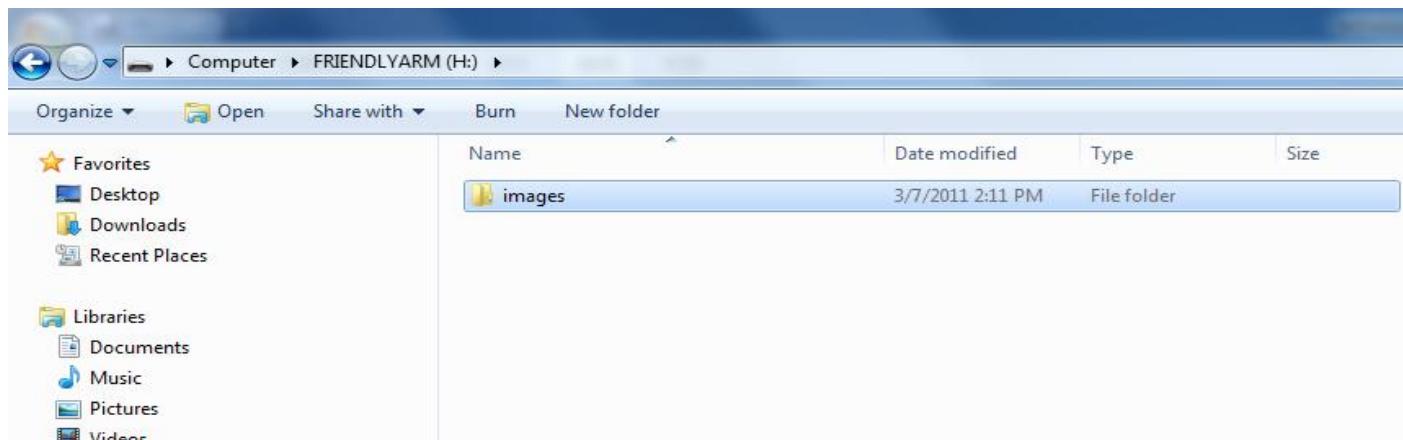
To run or install systems from an SD card which should have a Superboot some system files need to be copied to it. One of them is a configuration file named "FriendlyARM.ini". Let us experience rapid installation and running via some basic steps. The following steps are based on 4.3-inch systems. You need to adjust your steps for other systems accordingly.

2.3.1 Install Windows CE

Note: by default all boards are preinstalled with Linux. We will now install WindowsCE6.

Step1: copy the whole “images” directory into your SD card, double click to open the “images\FriendlyARM.ini” file, change “OS=Linux” to “OS=CE6”, save it and insert this card into your Mini6410-1405 board.

Note: after the whole images directory is copied into the SD card’s root directory it will look like this:



Step2: toggle S2 to the “SDBOOT” mode and insert your SD card

Step3: power on your board, you will hear a beep from the buzzer and LED4 will be flashing

Step4: within few seconds you will notice that LED3, 2 and 1 begin to flash one by one until two beeps are heard and then all LEDs are on and round robin flashing. This indicates that burning is done. The whole process takes less than 20 seconds.

Step5: toggle S2 to the “NAND” mode, reboot the system and you will see WindowsCE is loading.

2.3.2 Install Android

Android is now very hot and you may want to give it a shot! Let us try it:

Step1: copy the whole “images” directory into your SD card, double click to open the “images\FriendlyARM.ini” file.

Step2: change “OS=CE6” to “OS=Android”, save it and insert this card into your Mini6410-1405 board. Toggle S2 to the “SDBOOT” mode

Step3: power on your board, the installation process begins with a beep and ends with two beeps. The whole process takes less than 60 seconds.

Step4: toggle S2 to the “NAND” mode, reboot the system and you will see Android is loading.

Note: if you hear continuous beeps or see all four LEDs flash simultaneously it is an indication that your spelling in the ini file may be wrong.

2.3.3 Restore Linux

Step1: copy the whole “images” directory into your SD card, double click to open the “images\FriendlyARM.ini” file.

Step2: change “OS=Android” to “OS=Linux”, save it and insert this card into your

Mini6410-1405 board. Toggle S2 to the “SDBOOT” mode

Step3: power on your board, the installation process begins with a beep and ends with two beeps. Our Linux includes Qtopia-2.2.0, Qtopia4, QtE-4.7.0 and SMPlayer therefore the image file is relatively large. The whole process takes about 1 minute.

Step4: toggle S2 to the “NAND” mode, reboot the system and you will see Linux is loading.

Note: if you hear continuous beeps or see all four LEDs flash simultaneously it is an indication that your spelling in the ini file may be wrong.

2.3.5 FriendlyARM.ini

The “FriendlyARM.ini” file is a configuration file for system installation and running from SD card.

FriendlyARM.ini File

```
#This line cannot be removed. by FriendlyARM(www.arm9.net)
LCD-Mode = Yes
LCD-Type = N43

CheckOneButton=No
Action=install
OS= Linux
VerifyNandWrite=No
StatusType = Beeper| LED

##### Linux #####
Linux-BootLoader =superboot-6410.bin
Linux-Kernel = Linux/zImage
Linux-CommandLine = root=ubi0:FriendlyARM-root ubi.mtd=2 rootfstype=yaffs2 init=/linuxrc console=ttySAC0,115200
Linux-RootFs-InstallImage = Linux/rootfs_qtopia_qt4.img
Linux-RootFs-RunImage = Linux/rootfs_qtopia_qt4.ext3
```



WindowsCE6

WindowsCE6-Bootloader=superboot-6410.bin

WindowsCE6-BootLogo = WindowsCE6\bootlogo.bmp

WindowsCE6-InstallImage = WindowsCE6\NK-i.bin

WindowsCE6-RunImage = WindowsCE6\NK-i.bin

Android

Android-BootLoader =superboot-6410.bin

Android-Kernel = Android/azImage

Android-CommandLine = root=ubi0:FriendlyARM-root ubi.mtd=2 rootfstype=ubifs init=/linuxrc console=ttySAC0,115200

Android-RootFs-InstallImage = Android/rootfs_android-slc.ubi

Android-RootFs-RunImage = Android/rootfs_android.ext3

UserBin part

UserBin-Image=WindowsCE/NK-i.nb0

userBin-StartAddress=50100000

| Items | Note: different systems may have different default settings |
|-----------------|---|
| CheckOneButton | =“yes”, users need to press any button before the system can reboot completely after the system is reset, = “No”, system will reboot completely after it is reset. For mass burning this item is usually set to “No” |
| Action | Set actions: Install/Run/Null Install – Install to the NAND Flash Run – Run from SD card Null – No action The default option is “Install” |
| OS | Operating system to be loaded: Linux/WindowsCE6/Android/UserBin ; “UserBin” means standalone programs or single file image such as uCos2 and Rt-Thread. The default option is “Linux” |
| VerifyNandWrite | = “yes”, system will verify after burning is done. This is more reliable ; = “No”, system will not verify, this takes less time. The default option is “No” |
| StatusType | Status of the burning process: “LED”, “Beeper” and “LED Beeper” The default option is “LED Beeper” |



Linux' image settings, they can contain directories, “/” or “\”

| | |
|---|--|
| Linux-BootLoader | Bootloader's image: e.g. Linux-BootLoader=Linux/u-boot_nand-ram256.bin (by default) |
| Linux-Kernel | Linux kernel's image: e.g. Linux-BootLoader=Linux/zImage (by default) |
| Linux-CommandLine | Linux booting parameters, different file systems require differed settings for YAFFS2 recommended setting (this is also the default setting): Linux-CommandLine = root=/dev/mtdblock2 rootfstype=yaffs2 init=/linuxrc console=ttySAC0,115200 for UBIFS, recommended setting: Linux-CommandLine = root=ubi0:friendlyARM-root ubi.mtd=2 rootfstype=ubifs init=/linuxrc console=ttySAC0,115200 running from SD card, you can go with the default setting |
| Linux-RootFs-InstallImage | Image for installation: yaffs2 / UBIFS We define it as follows: The “img” extension is for yaffs2 images The “ubi” extention is for UBIFS images The “ext3” extention is for EXT3 images (only for running from the SD card) e.g. Linux-RootFs-RunImage=Linux/root-qtopia-qt4.img (by default) |
| Linux-RootFs-RunImage | Image to be run from the SD card, e.g. Linux-RootFs-RunImage=Linux/root-qtopia-qt4.ext3 (by default) |
| WindowsCE6's image settings, they can contain directories, “/” or “\” | |
| WindowsCE6-Bootloader | Bootloader's image: e.g. WindowsCE6\NBOOT_N43-RAM256.nb0 (by default) |
| WindowsCE6-BootLogo | Boot logo of WindowsCE6. This logo will be burned into the NAND Flash. It can be a bmp file and the max size is 2M e.g. WindowsCE6-BootLogo=WindowsCE6\BootLogo.bmp (by default) |
| WindowsCE6-InstallImage | Image for installation, it should be an NK.bin file e.g. : WindowsCE6-InstallImage=WindowsCE6\NK-i.bin (by default) |
| WindowsCE6-RunImage | Image to be run from the SD card, it should be an Nk.bin file, e.g. WindowsCE6-RunImage=WindowsCE6\NK-i.bin (by default) |
| Android's image settings, they can contain directories, “/” or “\” | |
| Android-BootLoader | Bootloader's image e.g. Android-BootLoader=Android/u-boot_nand-ram256.bin (by default) |
| Android-Kernel | Kernel image, e.g. Android-BootLoader=Android/azImage (by default) |
| Android-CommandLine | Android's booting parameters, different file systems require differed settings for yaffs2 recommended setting: Android-CommandLine = root=/dev/mtdblock2 rootfstype=yaffs2 init=/linuxrc |



| | |
|---|--|
| | console=ttySAC0,115200 for UBIFS recommended setting: Android-CommandLine = root=ubi0:friendlyarm-root ubi.mtd=2 rootfstype=ubifs init=/linuxrc console=ttySAC0,115200 To run from the SD card, users can go with the default setting |
| Android-RootFs-InstallImage | Image for installation: yaffs2/UBIFS We define it as follows: The “img” extension is for yaffs2 images The “ubi” extention is for UBIFS images The “ext3” extention is for EXT3 images (only for running from the SD card) e.g. Android-RootFs-InstallImage = Android/rootfs_android.ubi (by default) |
| Android-RootFs-RunImage | Image to be run from the SD card e.g. Android-RootFs-RunImage = Android/rootfs_android.ext3 (by default) |
| standalone program's image settings, they can contain directories, “/” or “\” | |
| Note: in general independent programs should have a RAM address for which it starts execution | |
| UserBin-Image | Standalone program's image, it can be a bin or nb0 file; when “Action” is “Install”, it will be burned into the initial address of the NAND Flash's Block0. |
| UserBin-StartAddress | When “Action” is “Run” the image will be loaded to the specified address of RAM |

Notes:

1. Statements after “#” will not be executed by Superboot. Actually any character except key words can be used to comment. Using “#” is just widely accepted
2. To prevent our Superboot from being illegally copied we make it a rule that the first line of the ini file cannot be edited or deleted. It is:

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

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

2.4 Introduction to Supertool

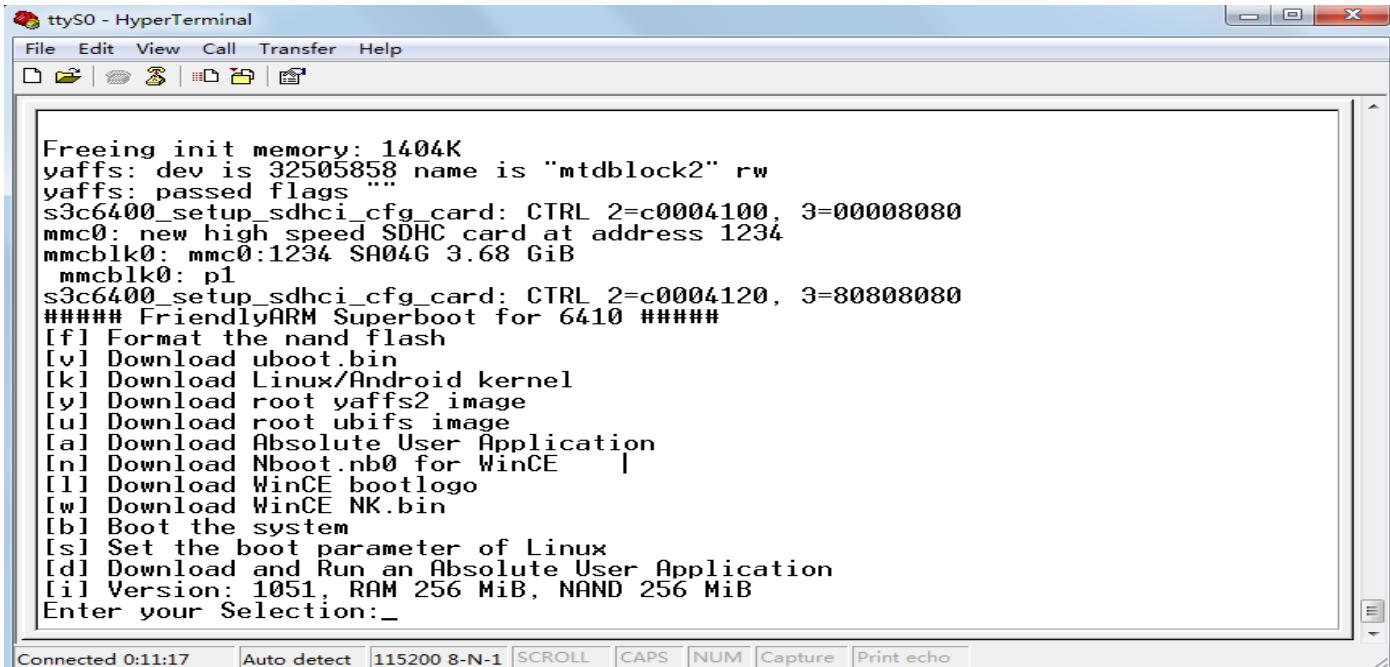
When using Superboot as the SD card's bootloader to boot the board, users will enter the USB download mode in the following two situations:

1. “images\FriendlyARM.ini” doesn't exist in the SD card or its file name is not spelled

correctly

2. When the ini file exists and “CheckOneButtons” is “Yes” users forget to press a key to continue the booting process.

In this mode the terminal will display the following menu and LED1 will be flashing continuously:



```

Freeing init memory: 1404K
yaffs: dev is 32505858 name is "mtdblock2" rw
yaffs: passed flags
s3c6400_setup_sdhci_cfg_card: CTRL 2=c0004100, 3=00008080
mmc0: new high speed SDHC card at address 1234
mmcbblk0: mmc0:1234 SA04G 3.68 GiB
mmcbblk0: p1
s3c6400_setup_sdhci_cfg_card: CTRL 2=c0004120, 3=80808080
##### FriendlyARM Superboot 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
[d] Download and Run an Absolute User Application
[i] Version: 1051, RAM 256 MiB, NAND 256 MiB
Enter your Selection:_

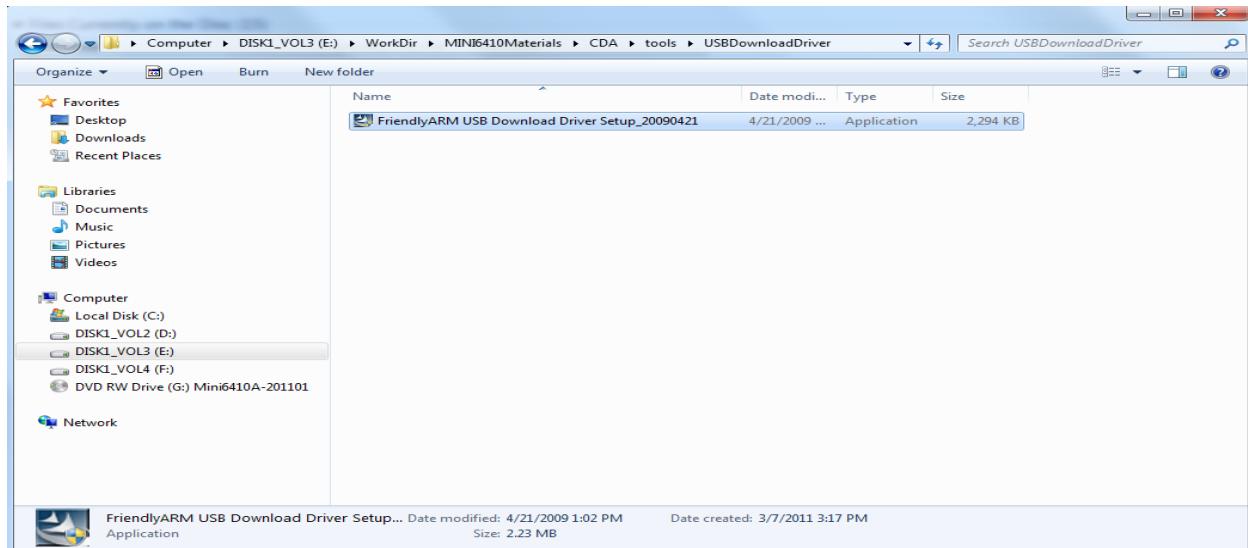
```

Connected 0:11:17 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

2.4.1 Install USB Download Driver

Installing this USB driver doesn't need to connect to a board. It is just for the PC system. It works in Windows7 but not in 64-bit Windows7.

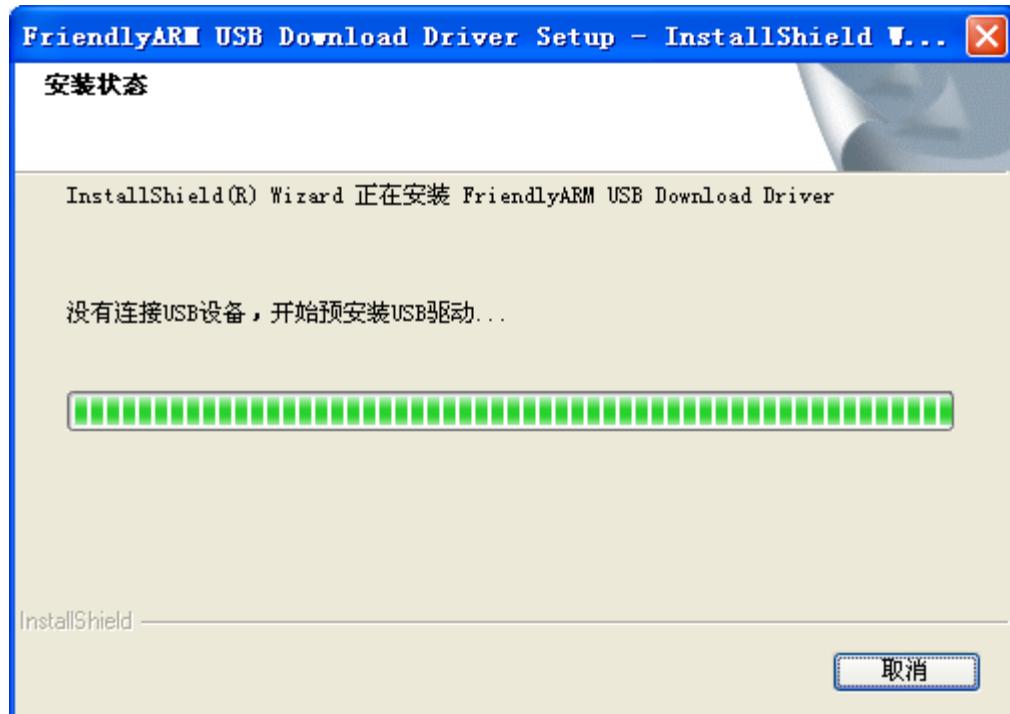
Open the shipped CD, double click on “**tools\USBDownloadDriver\ FriendlyARM USB Download Driver Setup_20090421.exe**” “**WindowsPlatformTools\USBDownloadDriver\FriendlyARM USB Download Driver Setup_20090421.exe**” to start installation



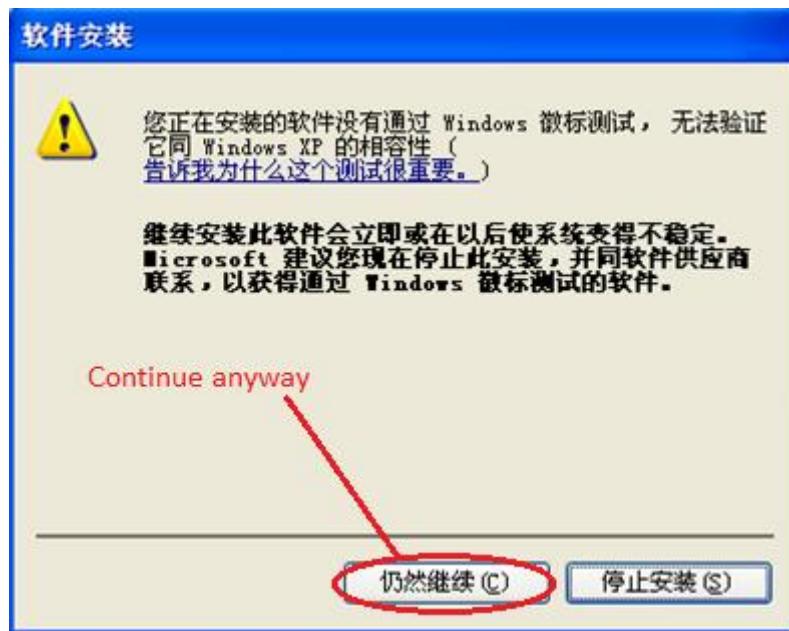
The following window will show up:



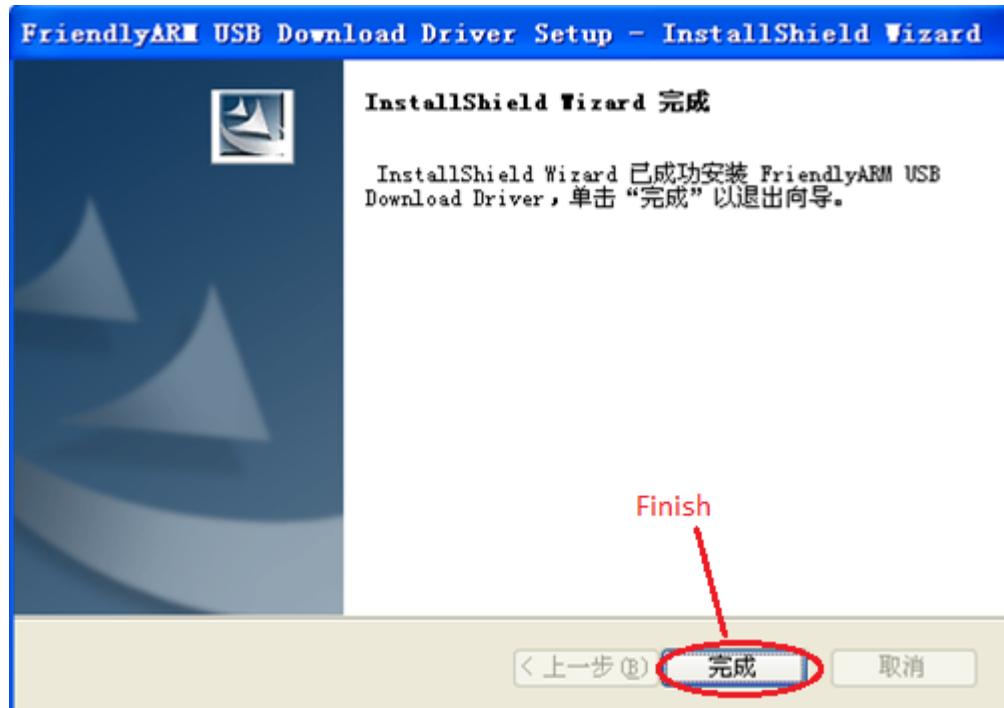
Click on (“Next”) the middle button



A warning message will pop up



Click on the (“continue anyway”) the left button to finish the installation.



Now let's test the USB driver:

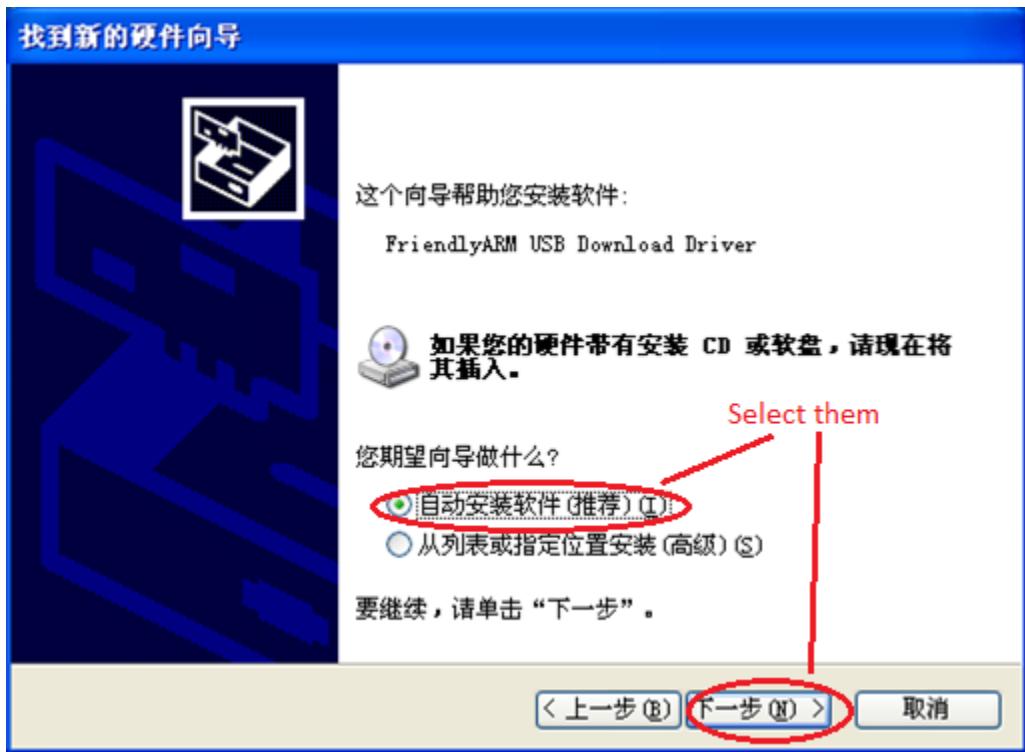
Connect the Mini6410-1405 board to a host PC via a USB cable. Toggle the S2 switch to the “SDBOOT” side.

Turn on the S1 switch, if this is the first time you connect, Windows XP will prompt that a new USB device is found. Follow the steps below to install a USB driver

(1) After the following window pops up, check the third option and click on the “Next” button



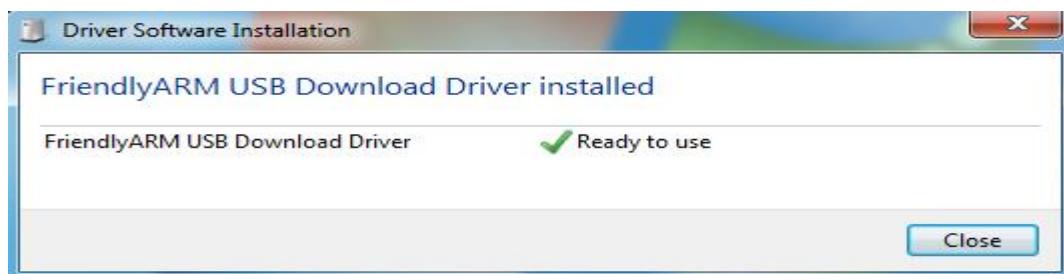
(2) On the window shown below, check the first option and click on the “Next” button



On the following popup window, click on the left button (“Continue anyway”).



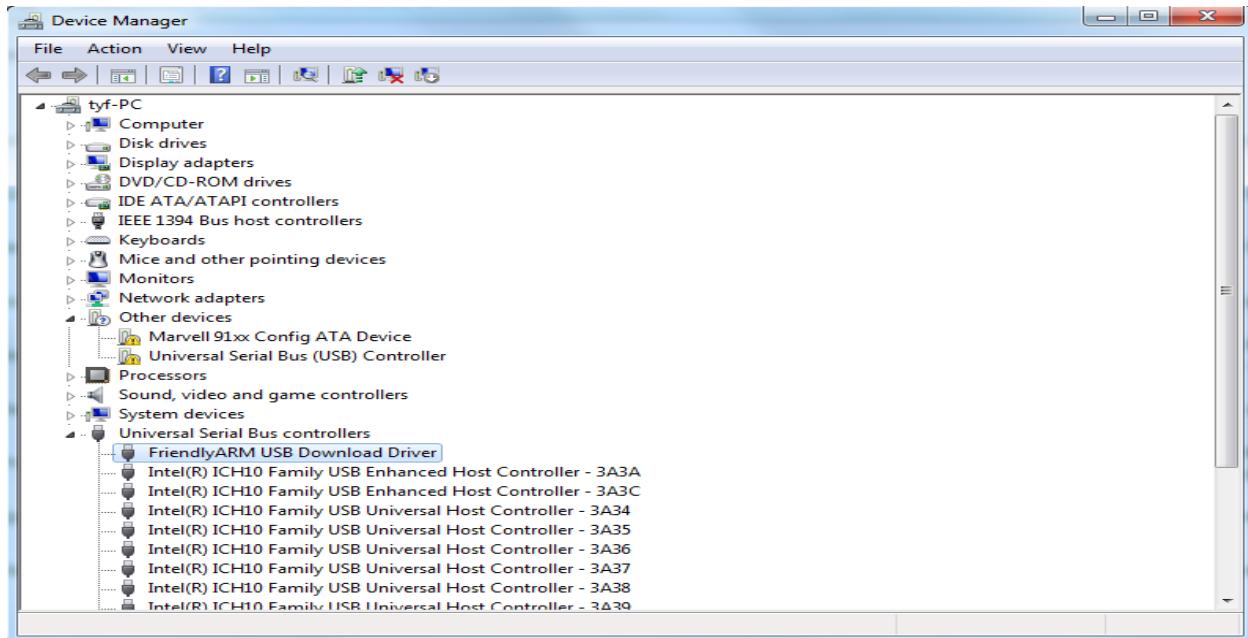
OK, our installation is done.



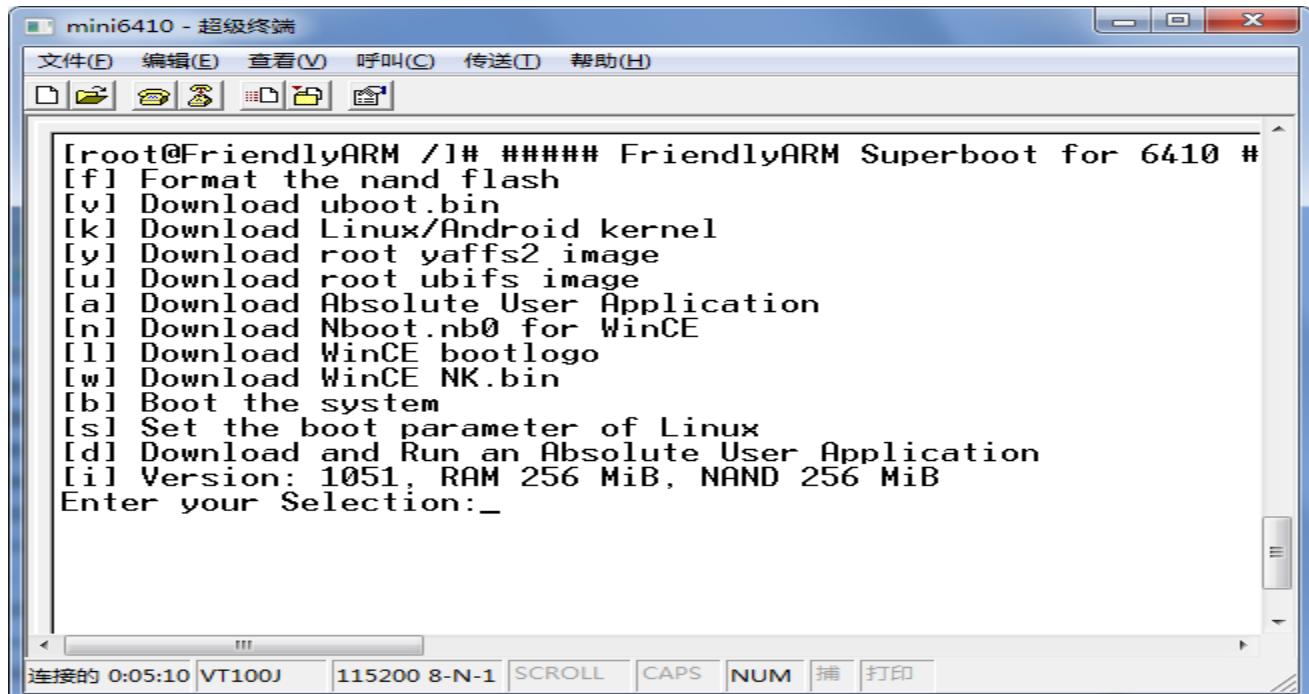
Open the CD, click on the dnw.exe, if you can see “USB:OK”, this means the installation is a success



In the device manager, you will see the installed USB driver information:



2.4.2 Superboot's Menu



```
[root@FriendlyARM /]# ##### FriendlyARM Superboot 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
[d] Download and Run an Absolute User Application
[i] Version: 1051, RAM 256 MiB, NAND 256 MiB
Enter your Selection:_

连接的 0:05:10 VT100J 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

Note: the above menu may subject to changes.

Item[f]: Format the Nand Flash. This command will delete all the data in it

Item[v]: Download a linux bootloader e.g. U-boot

Item[k]: Download a linux/Android kernel

Item[y]: Download an image of the yaffs2 file system

Item[u]: Download an image of the UBIFS file system

Item[a]: Download an Absolute User Program(standalone), usually it is a bin file, such as uCos2.

Item[n]: Download a WinCE's bootloader Nboot

Item[l]: Download a WinCE's boot logo

Item[w]: Download an image of WinCE NK.bin

Item[b]: Boot system, if the board is installed with either WinCE or Linux, it will load it

Item[s]: Set boot parameters

Item[d]: Download and run an Absolute User Program(standalone program)

Item[i]: Display Superboot's version and the NAND Flash's memory

Chapter 3 Install, Upgrade and Run Systems

Friendly ARM creates the way of installing operating systems via USB download for the Mini2440 board. We advanced this technology further for our Mini6410-1405 by exploring its feature of supporting booting from the SD card. For instance, we developed a bootloader that supports FAT32. This way users can read image files directly from the SD card without going via USB download from a PC.

If you have stepped through our previous chapters you would have experienced this new way of installation. It is prompt and easy. In our upcoming products we will deliver more cool features.

A lot of our customers have already been used to installing systems via USB download especially for development and testing. Therefore we kept this feature in our Mini6410-1405. The required utilities are identical to those for the Mini2440.

We will start this section by introducing the “one key installation” feature.

3.1 Install Systems with Minitools

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

- Only need a USB cable: with the Minitools users only need a USB cable to install systems
- One key action: no need to type any command.

- Works with both 32/64-bit OS: it can be installed on both 32-bit and 64-bit Windows systems
- Cross platform: it can be installed on both Windows and Linux systems

3.1.1 Install Minitools

3.1.1.1 Install on Windows

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

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



The minitools’ main window is shown below:



3.1.1.2 Install on Linux

We tested installing the Minitools on Fedora9/Fedora15/Ubuntu12.04 64-bit systems. Please login and execute the installation as root. Please copy the “Minitools-Linux-YYYYMMDD.tgz” in the “tools” directory from your DVD to your PC and untar the ball and run the “./start.sh” command to the installation.

3.1.2 Flash Superboot to SD Card

In order to work with the Minitools you need to get an SD card and flash our superboot to it. Please follow the steps below:

1. Please flash the superboot to an SD card with “SD-Flasher”
2. Please copy the whole “images” directory from your DVD to the root directory of your

SD card

3. Open the “images/friendlyARM.ini” and add the following line

USB-Mode = yes

Please follow the steps below to connect your board to your PC

1. Switch the S2 on your board to “SD”
2. Power on the board and you will see the LCD showing “USB Mode:Waiting” if everything works correctly
3. Please connect your board to your PC via a USB cable
4. If the connection is successful the LCD will show “USB Mode:Connected”

Now you can start installing systems with the Minitools

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

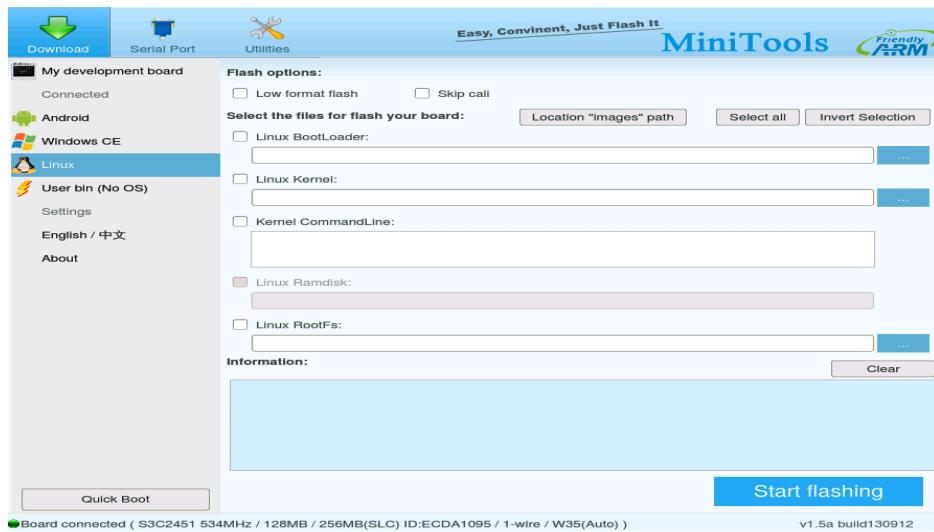
3.1.3 Install Systems with Minitools

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

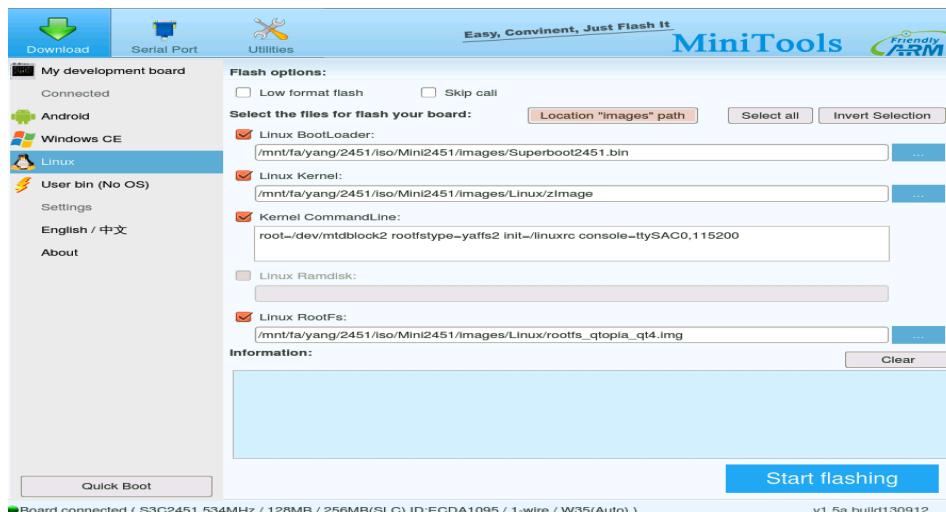


On the left bottom of the window there is an LED which is green indicating the board is connected successfully. On the left bottom there is a button which can start your board directly without switching to NAND.

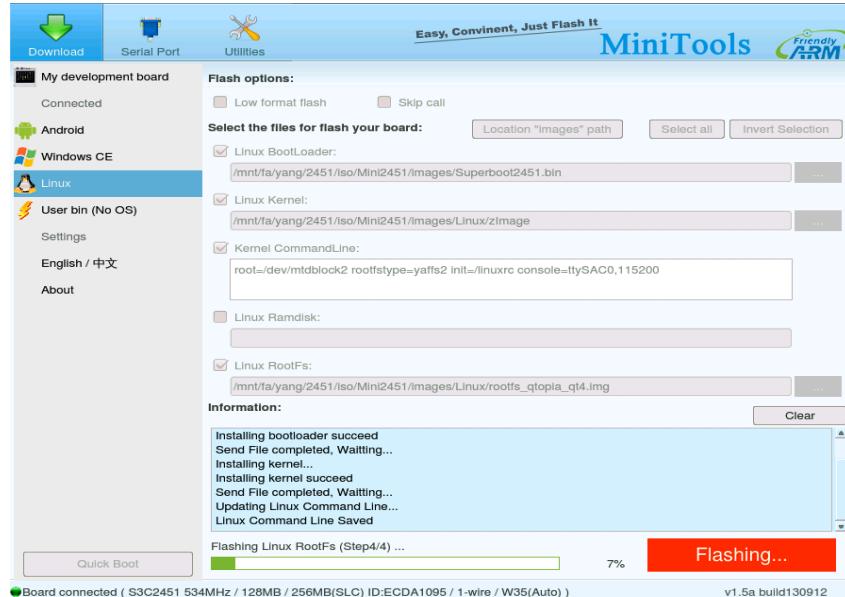
Before install systems please select the system you want to install e.g. Linux and then its configuration will be presented as follows:



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

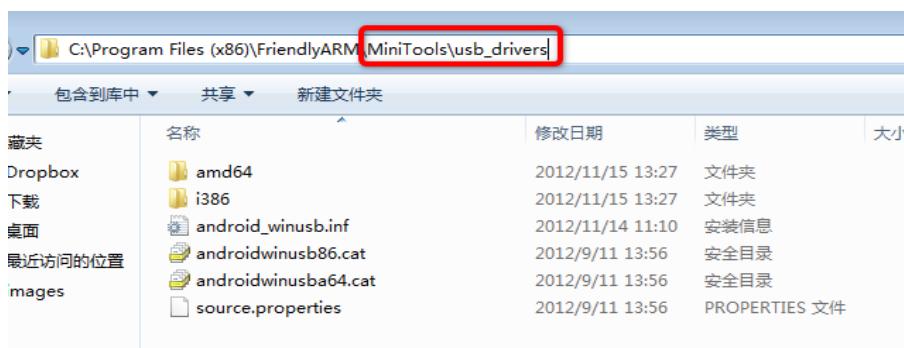


With the Minitools utility you can update either the whole system (all image files) or individual image files e.g. the kernel image file. After you are done with your installation configuration please click on “Start flashing”



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

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



3.2 Install System from SD Card

To install systems from the SD card users need to use our SD-Flasher utility to burn a Superboot into the SD card (see 2.2) and copy related system files to its images directory. Those files are in the images directory in the shipped CD. If you want to use your own files you can just copy your files into that directory.

Superboot supports both a common SD card and a high speed large memory card. We will use the files in our shipped CD to show you how to install systems.

Note: you can change your configurations in the FriendlyARM.ini file in the following steps based on your preferences or use the one in our CD (CDB\images\)

3.2.1 Install Linux (YAFFS2)

Note: YAFFS2 only applies to SLC Nand Flash

Step1: open the FriendlyARM.ini file in the SD card's images directory and make changes as follows:

| Items | Options (case insensitive) |
|--|---|
| Action | Install |
| OS | Linux |
| Linux-Kernel | Linux/zImage (or your own image) |
| Linux-CommandLine | root=/dev/mtdblock2 rootfstype=yaffs2 init=/linuxrc console=ttySAC0,115200 |
| Linux-RootFs-InstallImage | linux/rootfs_qtopia_bt4.img |
| Note: Superboot will automatically detect the LCD type after installation begins | |

Step2: toggle the S2 switch to “SDBOOT” and insert an SD card

Step3: power on and you will hear a beep and LED4 begins to flash

Step4: Within seconds you will see that LED3, 2 and 1 begin to flash one by one and finally you will hear two beeps and all LEDs are on and round-robin flashing. The system is completely up and running

3.2.2 Install WindowsCE6

The following steps for 4.3" LCD systems

Step1: open the FriendlyARM.ini file in the SD card's images directory and make changes as follows:

| Items | Options (case insensitive) |
|-------------------------|--|
| Action | Install |
| OS | WindowsCE6 (alternatives: "CE6" or "Wince6") |
| WindowsCE6-Bootloader | WindowsCE6\superboot-6410.bin |
| WindowsCE6-BootLogo | WindowsCE6\bootlogo.bmp (or your own bmp) |
| WindowsCE6-InstallImage | WindowsCE6\NK-i.bin (or your own image) |
| Note: | |

Step2: toggle the S2 switch to "SDBOOT" and insert an SD card

Step3: power on and you will hear a beep and LED4 begins to flash

Step4: Within seconds you will notice that LED3, 2 and 1 begin to flash one by one and finally you will hear two beeps and all LEDs are on and round-robin flashing. The system is completely up and running

3.2.3 Install Android (YAFFS2)

Note: YAFFS2 only applies to SLC nand flash for Android

The following steps for 4.3" LCD systems

Step1: open the FriendlyARM.ini file in the SD card's images directory and make changes as follows:

| Items | Options (case insensitive) |
|-----------------------------|--|
| Action | Install |
| OS | Android |
| Android-Kernel | Android/azImage(or your own image) |
| Android-CommandLine | root=/dev/mtdblock2 rootfstype=yaffs2 init=/linuxrc console=ttySAC0,115200 |
| Android-RootFs-InstallImage | Android/rootfs_android.img |

Note: words in red should be typed exactly like what are presented here.

Step2: toggle the S2 switch to "SDBOOT" and insert an SD card

Step3: power on and you will hear a beep and LED4 begins to flash

Step4: Within seconds you will notice that LED3, 2 and 1 begin to flash one by one and finally you will hear two beeps and all LEDs are on and round-robin flashing. The system is completely up and running

Chapter 4 Explore Linux

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-4.8.5 are extremely user friendly. We would be glad to see our Mini6410-1405 being your gateway to the wonderful world of embedded system development and learning.

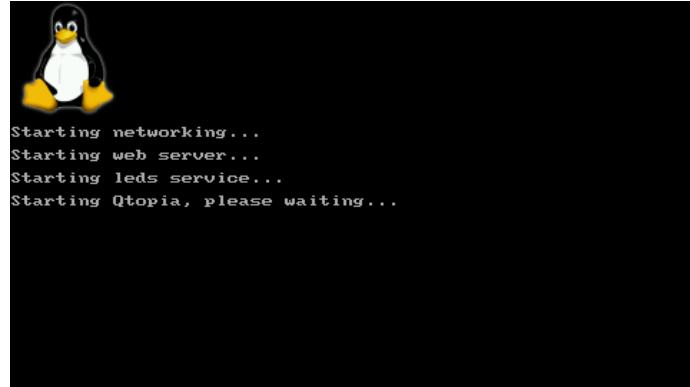
4.1 Get Started with Qtopia-2.2.0 and QtE-4.8.5

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.

To get the latest QtE please visit <http://qt.nokia.com/>. The version utilized in our system is QtE-4.8.5.

For most of our released systems, we have installed Linux + Qtopia 2.2.0+Qtopia4+QtE-4.8.5+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.

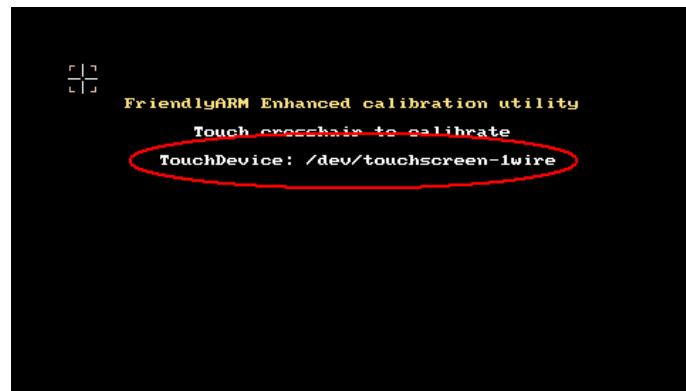


4.1.1 Calibrate Touch Screen

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.

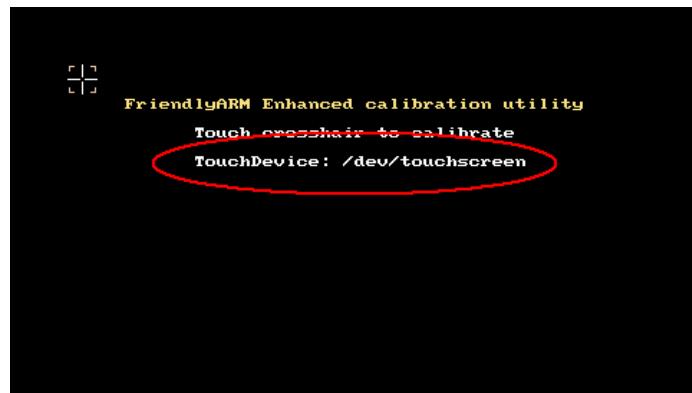
You will see the calibration interface under the following two situations:

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 touching

device:/dev/touchscreen-1wire, if the ARM system has an integrated touch screen interface it will be “/dev/touchscreen”

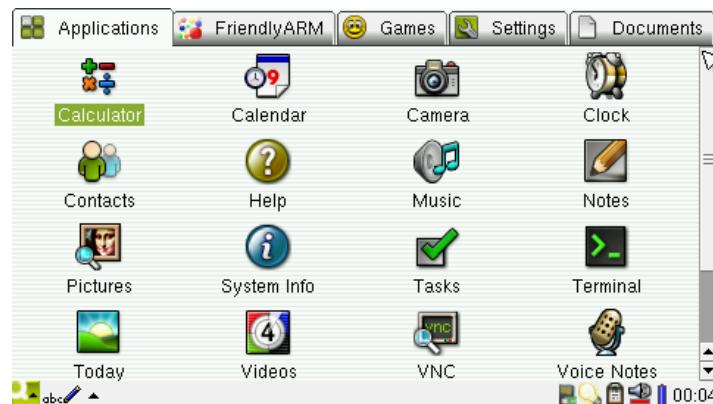


2. After entering the system, go to “Start” -> “Settings” -> “Configurations” -> “Recalibrate”.
Click on the “+” signal.



4.1.2 Main Window

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.



4.1.3 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 Mini6410-1405 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 Mini6410-1405, which is based on Qt4.x libraries 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-2.6.36 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.

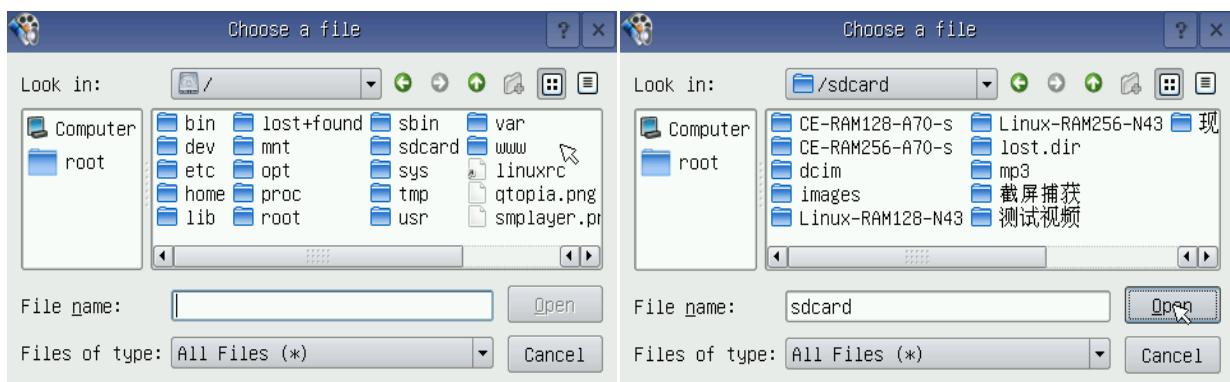
In our shipped CDs there are several test video files for testing.

4.1.3.1 Play Video with SMPlayer

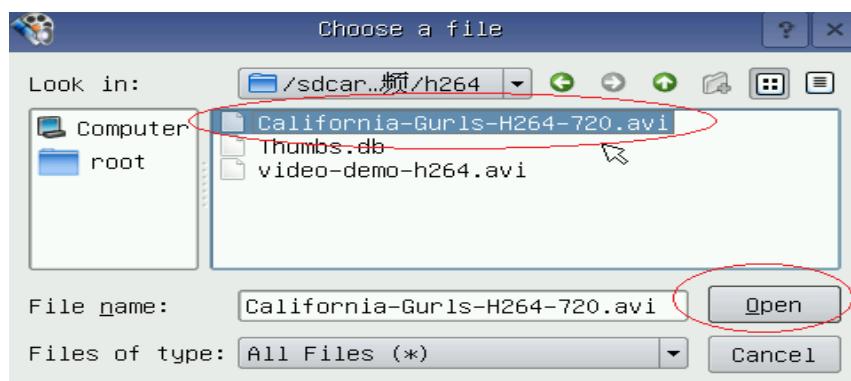
In the “Friendly ARM” tab, click on “SMPlayer”



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 “Friendly ARM” 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.



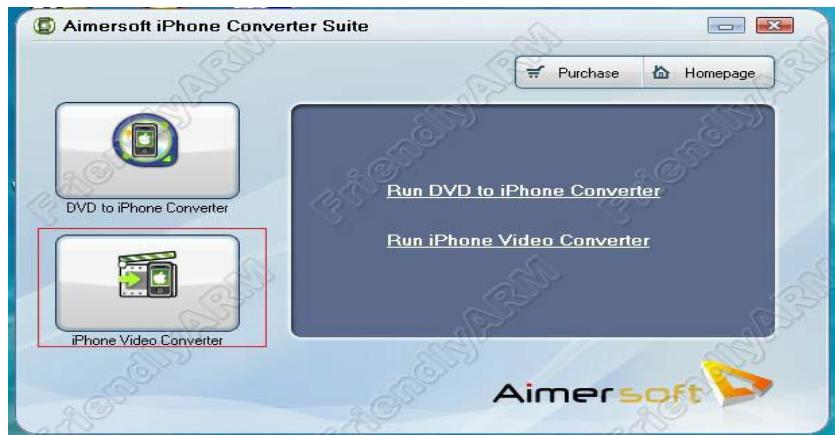
4.1.3.2 Convert Video Files

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 Mini6410-1405. 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 versions 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

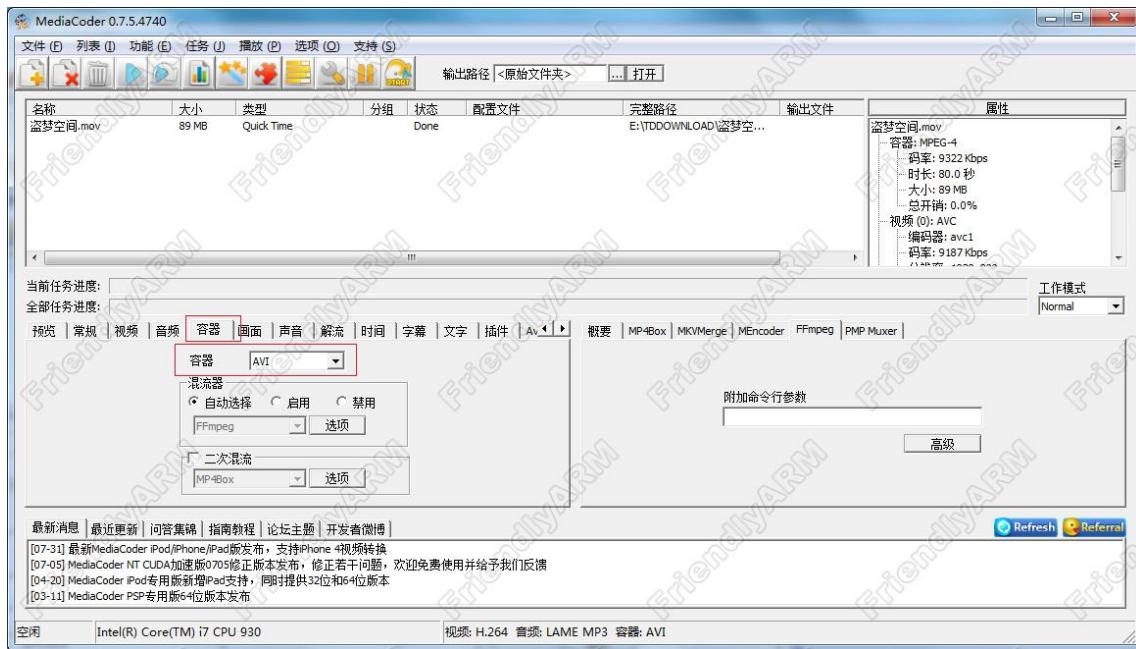
Convert to H264:

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/>.

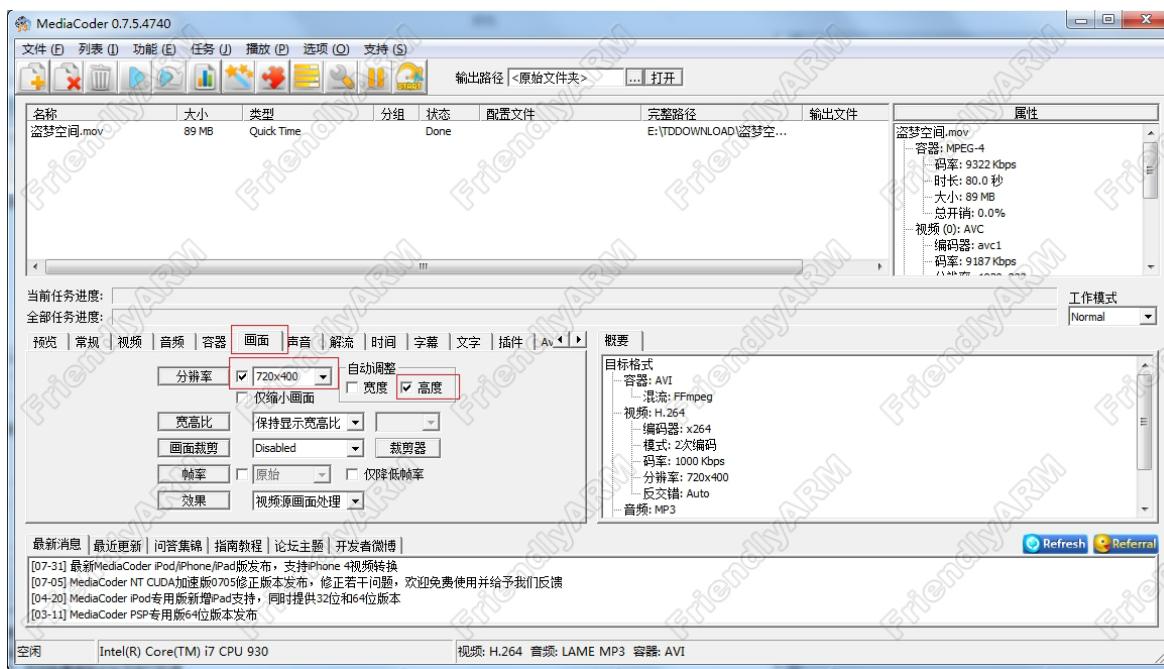
Start MediaCoder and you will see its main menu as below. Click on “+” on the upper left corner and follow the settings marked in red to configure



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.

4.1.4 WiFi AP

To start the WIFI AP function you need to use the RT8192 WiFi module.

Note: the router function hasn't been implemented.

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



Click on the the icon you will see the following window:



The fields on the window are defined as follows:

| | |
|--------------------|---|
| IP | IP Address |
| SSID | Access point |
| Password | Password, at least 8 digits with WPA2 encoding |
| Channel | The default value is 8. Note: if your working area has a wireless router this value cannot be the same as the router's. Usually most popular routers such as D-LINK's value is 6. |
| Opt Mode | WiFi protocol. The default is 802.11g |
| Auto start at boot | Whether to set auto start the WiFi AP |

After configuration is done click on “Apply” to save. Click on “Start” to start the WiFi AP. When it shows “Working” it means the board is now working under the WiFi AP mode. In our example the IP was 192.168.2.1

Status: Working (192.168.2.1)

Now you can find the board from your mobile phone. In our example the SSID was “arm9.net” we could find it on our mobile phone. Click on it and input the password to connect.



After the connection is successful you will find similar information shown as below



In our example the board's IP was 192.168.2.1 we could browse <http://192.168.2.1> on our mobile phone.



WiFi AP Setting will save WiFi's configuration data to the following files:

| Configuration File | Comment |
|--------------------------|--|
| /etc/hostapd.conf | WiFi parameters |
| /etc/hostcpd.conf | DHCP service parameters |
| /etc/rc.d/init.d/wifiapd | Start/Stop WiFi AP service |
| /etc/init.d/rcS | If you want to auto start WiFi AP service you can add “/etc/rc.d/init.d/wifiapd start” |

4.1.5 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

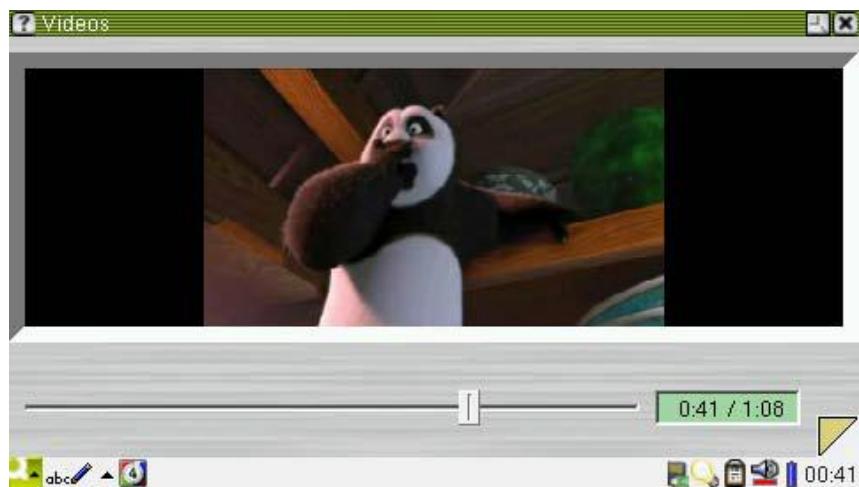
“Documents” tab and play it there.



4.1.6 Play Video

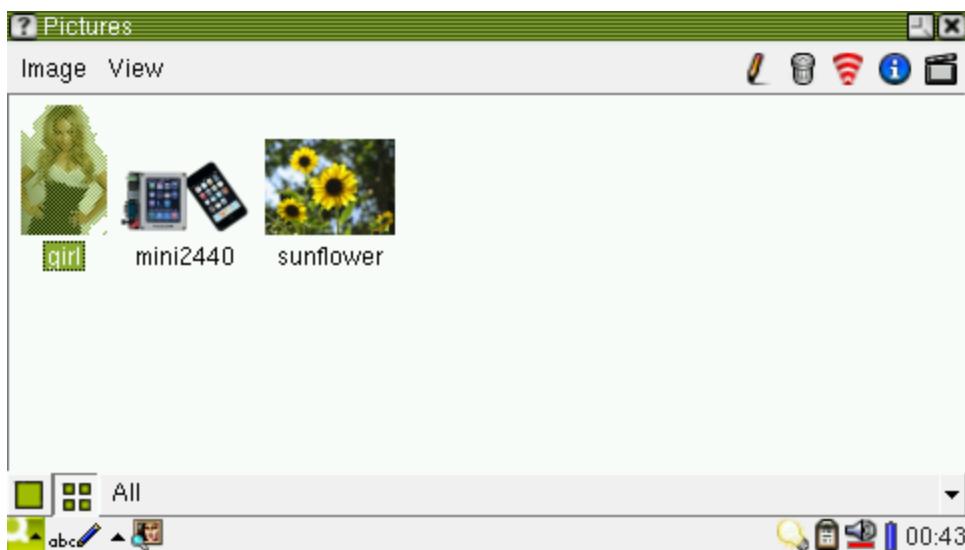
Go to the “Applications” tab and click on “Video”. In the “Video” list you can select a video file and play. This player is integrated in Qtopia it can only do soft decoding thus couldn’t play H.264/H.263/Mpeg4.

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



4.1.7 View Pictures

Go to the “Applications” tab and click on “Pictures”. You will see icons in the “Documents” group. If you have an 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.

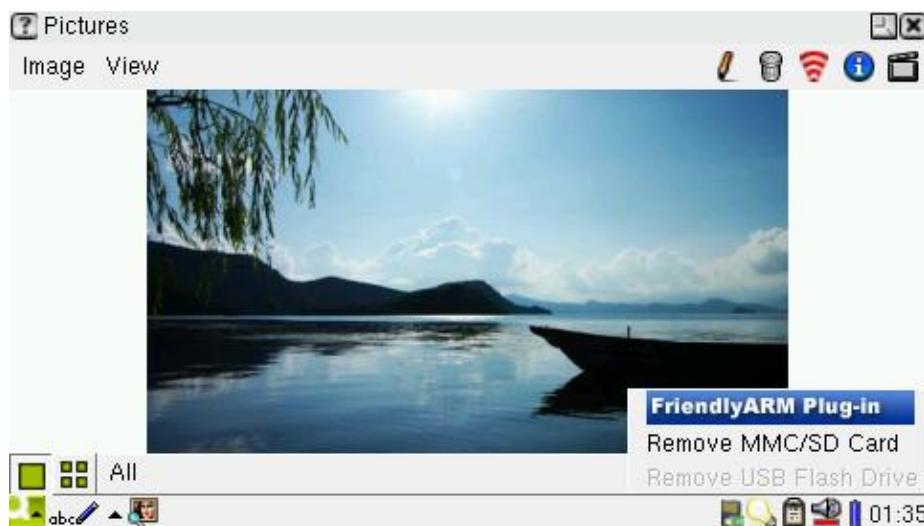
4.1.8 Auto Mount SD Card/Flash Drive

Insert a common or high speed SD card (max memory 32G) or a USB drive moments later a small icon will pop up on the lower right of the screen. The Mini6410-1405 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

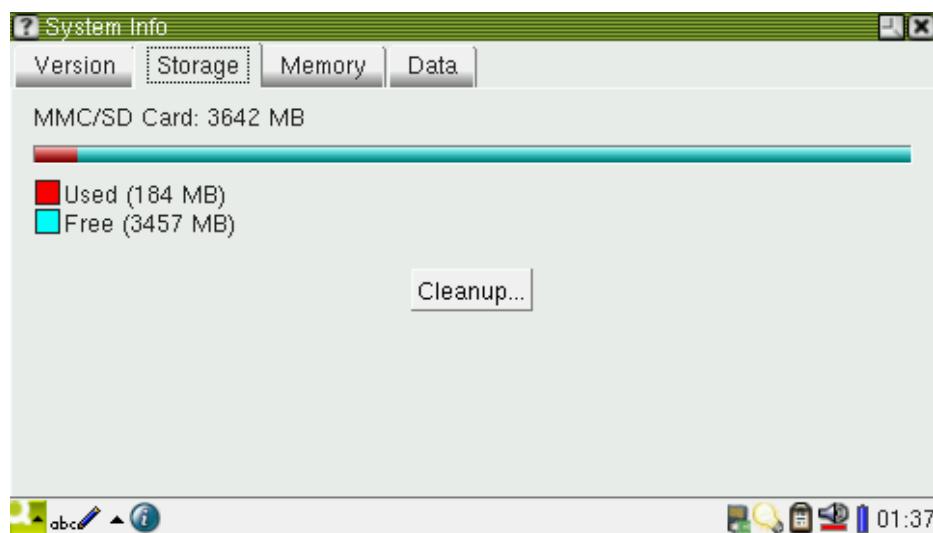
All files in the MMC/SD card or the flash drive can be viewed in the “Documents” tab.

Their directories will not be displayed.

Auto mounting of a MMC/SD card or a flash drive is developed by Friendly ARM 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:



4.1.9 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”.



4.1.10 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:

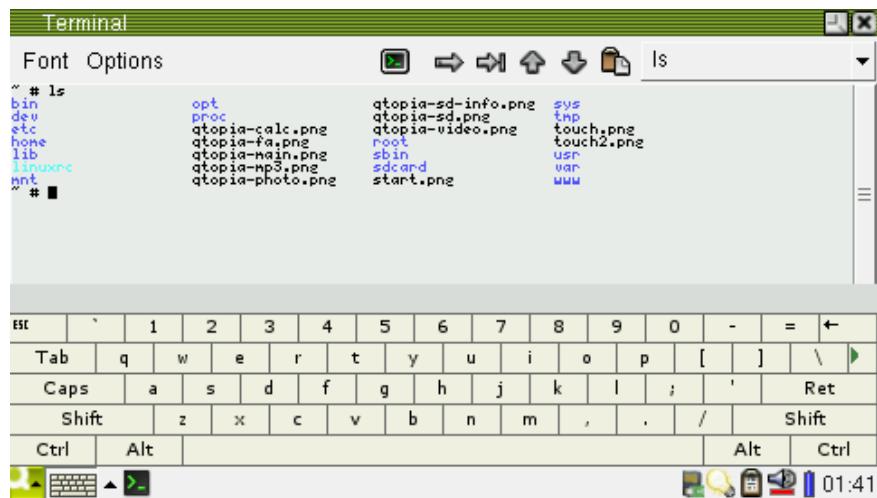
On system startup, if a terminal is bound to the serial port all its outputs and inputs are to and from this serial port. This is a common Linux way.

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 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 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.



4.1.11 File Manager

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

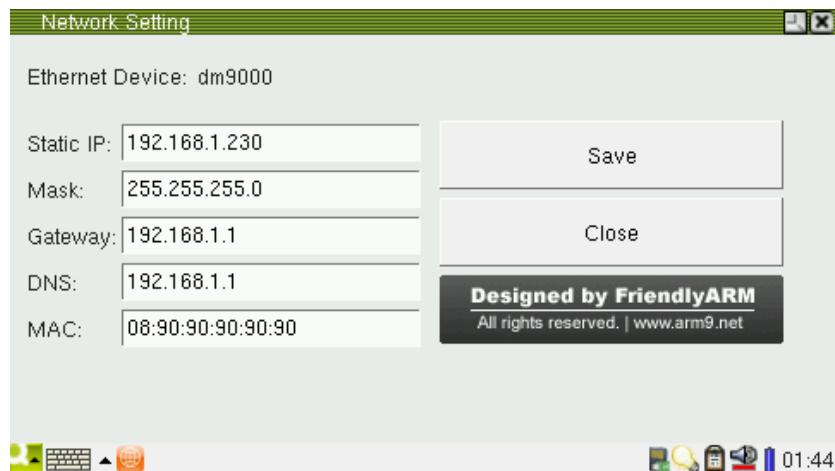


You can browse and manage files and directories

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.

4.1.12 Ethernet Configuration

Go to the “Friendly ARM” 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.

4.1.13 Setup WiFi

This section will guide you through the steps to set up WiFi. The 6410 supports most of the popular USB WiFi cards:



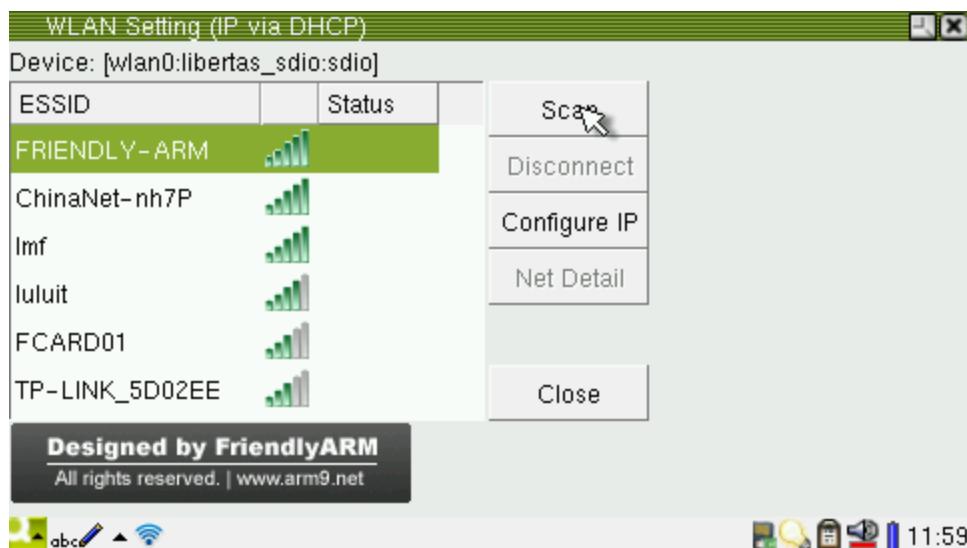
4.1.13.1 Launch WiFi Utility

Go to the “Friendly ARM” tab and click on “WLAN Setting”

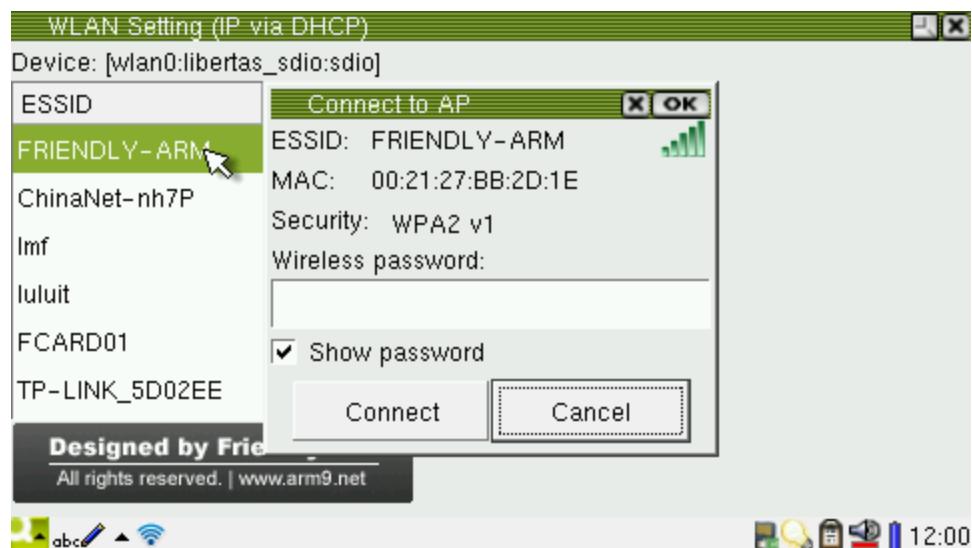


4.1.13.2 Connect to Wireless AP

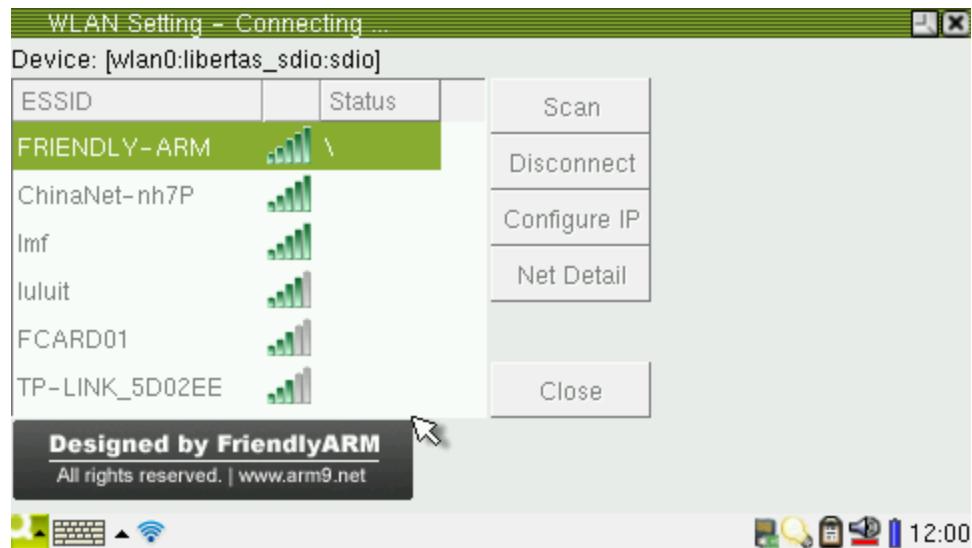
After the WLAN Setting utility is started it will automatically search for wireless AP and list their SSIDs and signal strength. If your utility doesn't show sources please click on “Scan”



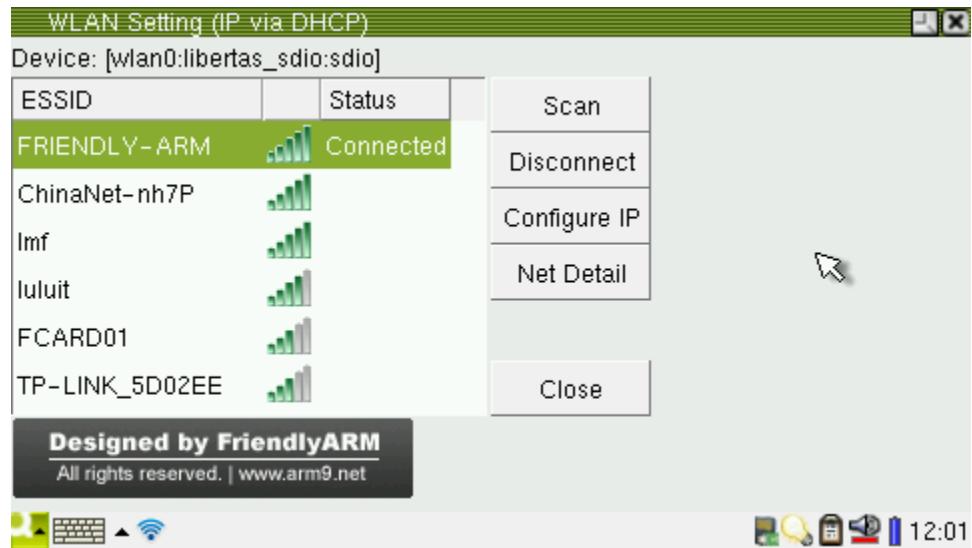
After an AP is found click on its SSID to connect. The following dialog will pop up and you need to input its password:



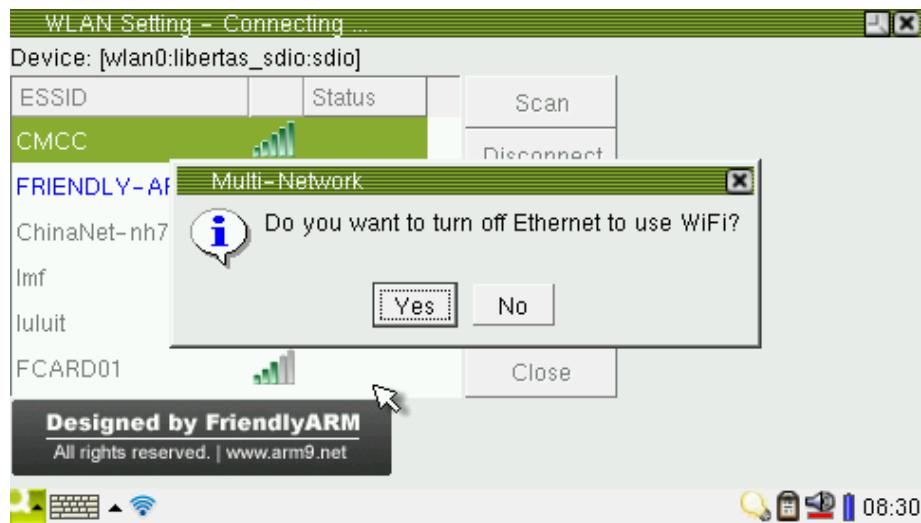
After typing the password (leaving it as blank if no password is set) click on “Connect”



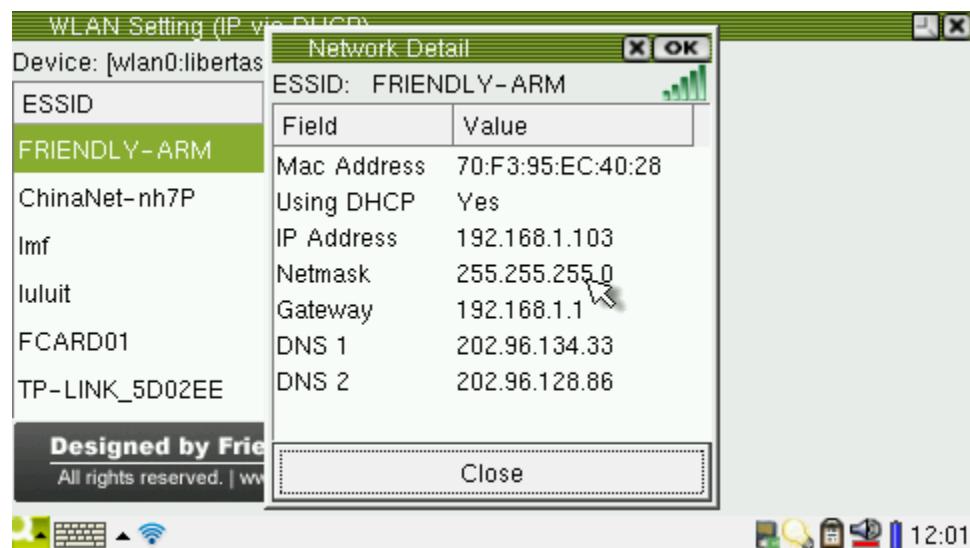
If the connection is successful, its status will show “connected”



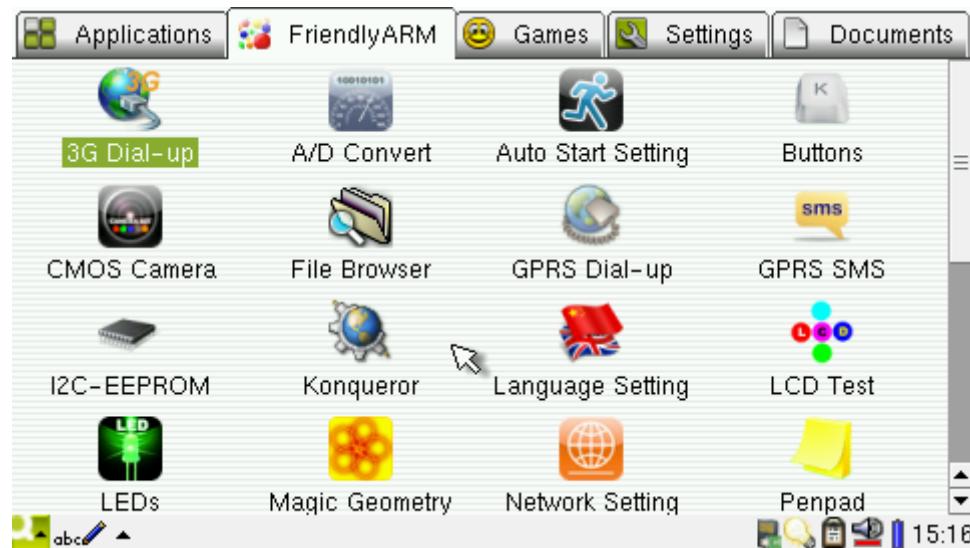
If your Ethernet is connected the following dialog may pop up asking you to disconnect it (ifconfig eth0 down) otherwise some network utilities would connect the Ethernet rather than the WiFi. Click on “Yes” to close it. To reconnect it you can either start “Network Setting” or type “ifconfig eth0 up”.



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

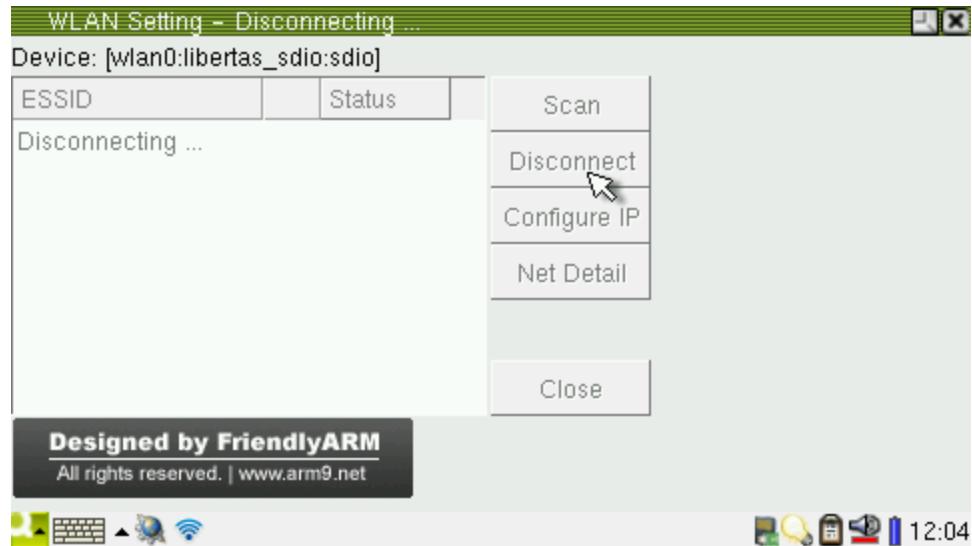


Now you can try your WLAN



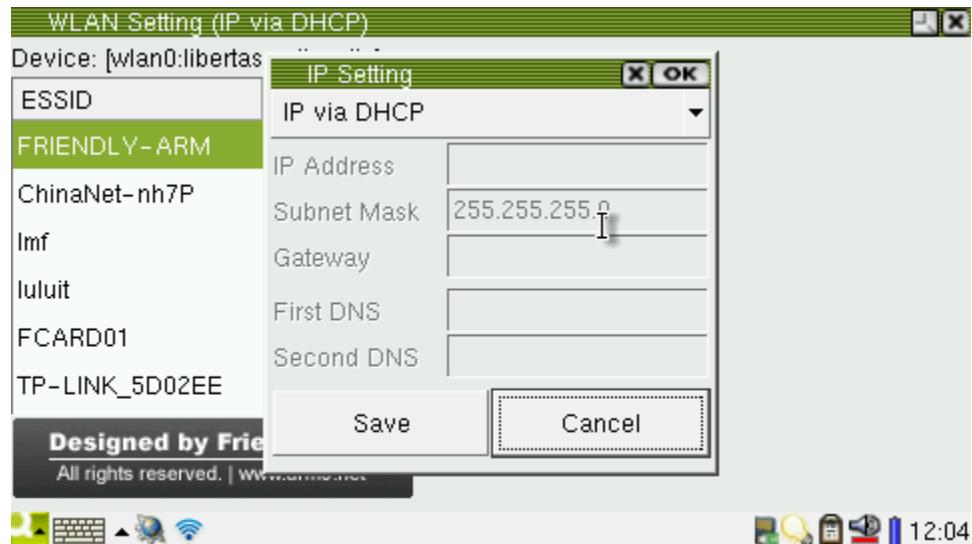
4.1.13.3 Disconnect WiFi

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

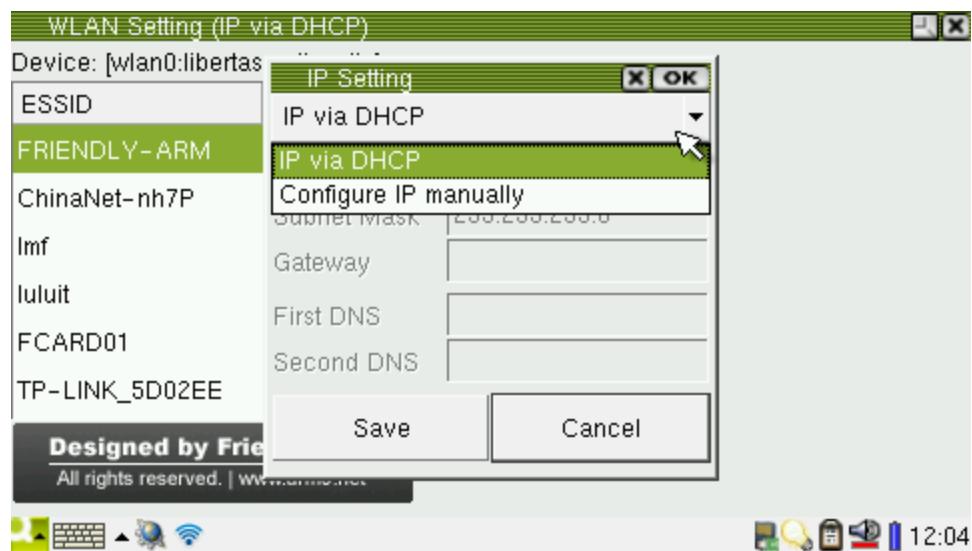


4.1.13.4 Configure WiFi IP

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”

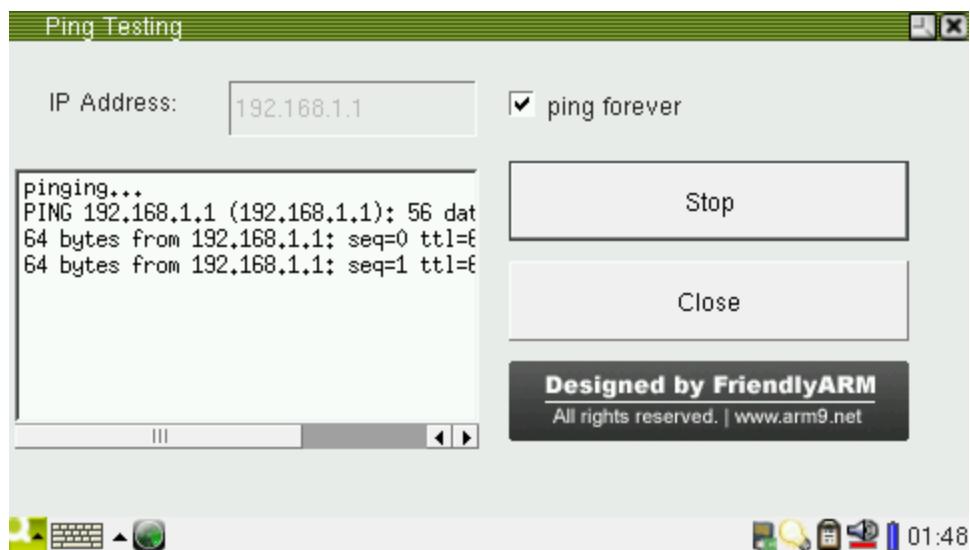


After configuration is done, click on “Save” to keep your settings

4.1.14 Ping Test

Connect your board to a network, set up your network and you will be able to test PING.

Go to the “Friendly ARM” 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 is 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.

4.1.15 Web Browser

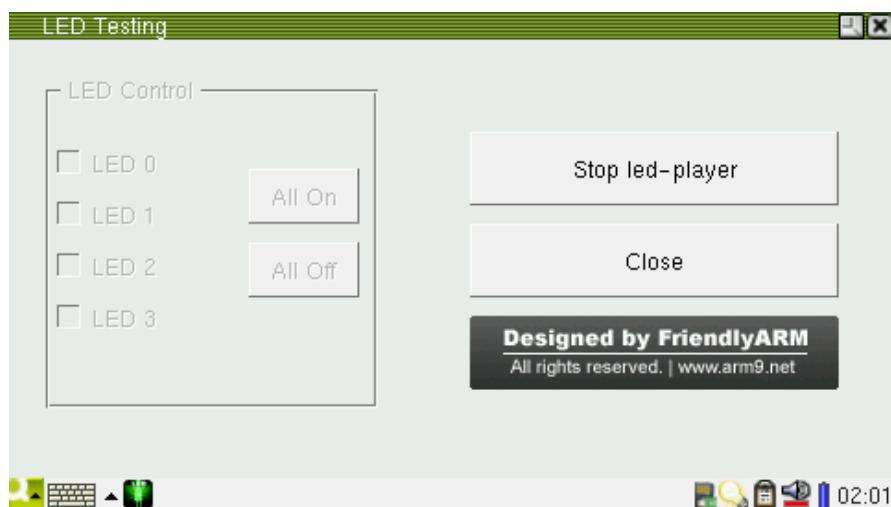
Go to the “Friendly ARM” tab and click on “Browser”. Click to start the “soft keyboard” on the lower right of the screen, type a website and click on “Ret” on the keyboard you will be able to visit your website.

Note: the browser utilized in the 6410 is Konqueror/Embedded, which is open source



4.1.16 LED Test

Go to the “Friendly ARM” 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.

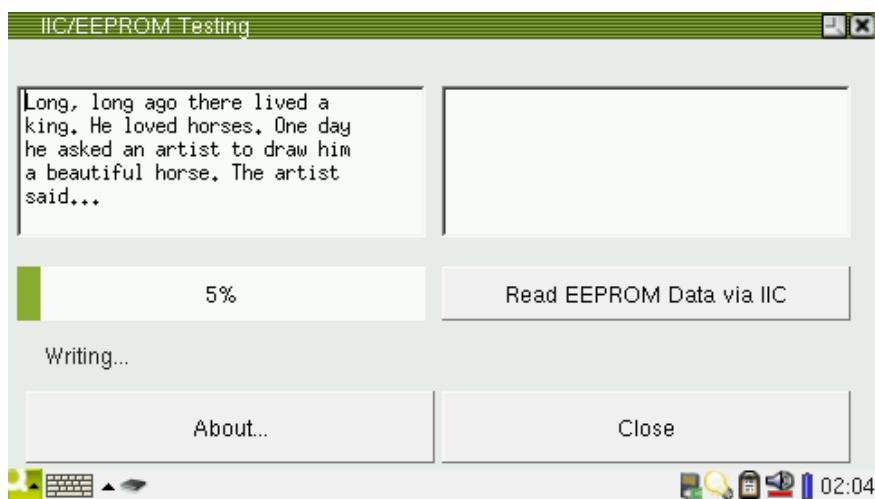
When you close the LED Testing GUI the led-player service will be restarted.



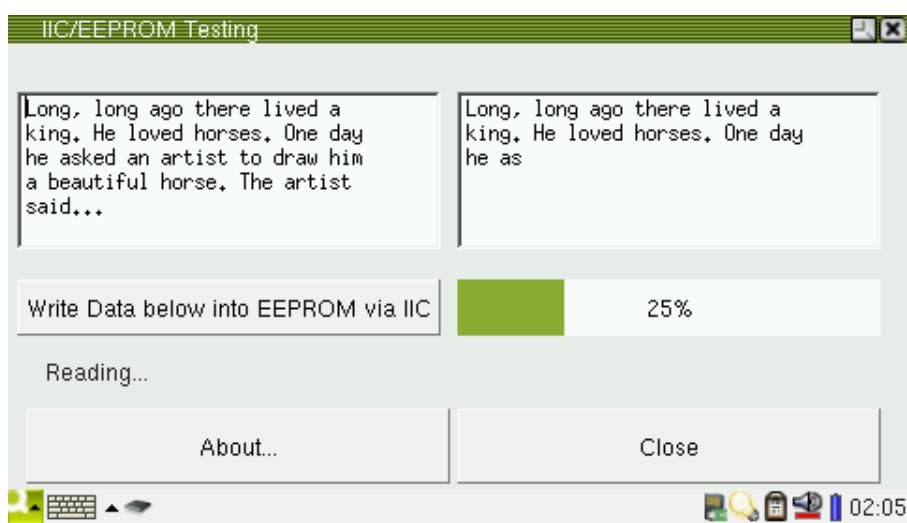
4.1.17 EEPROM

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

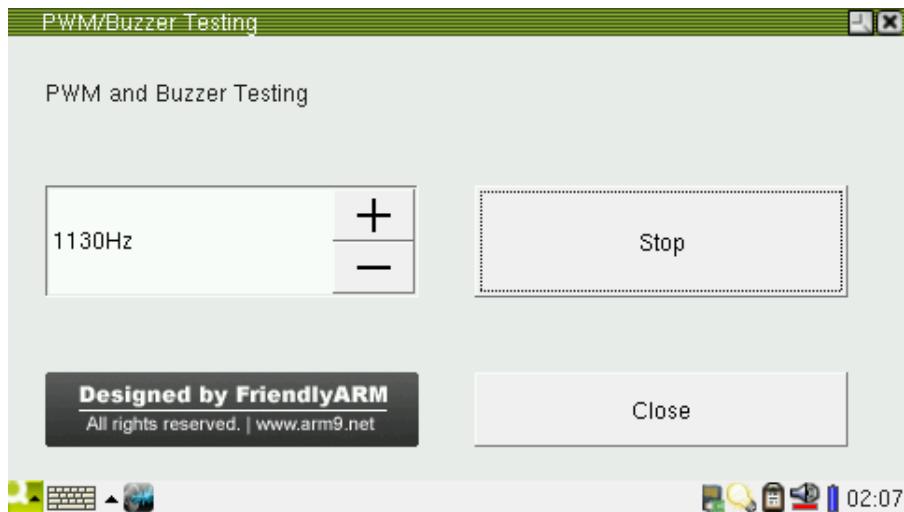


4.1.18 PWM Buzzer

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



4.1 19 Serial Port Assistant

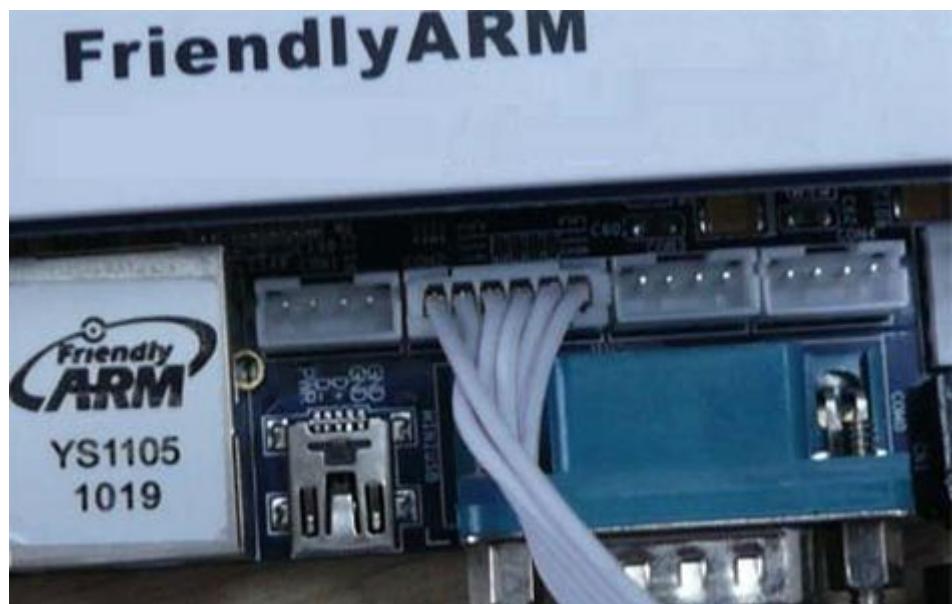
Note: before start this program please connect the serial port your want to test to your board.

- 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 use a USB hub to extend your serial ports.



Connect your serial port extension board to the Mini6410-1405'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.



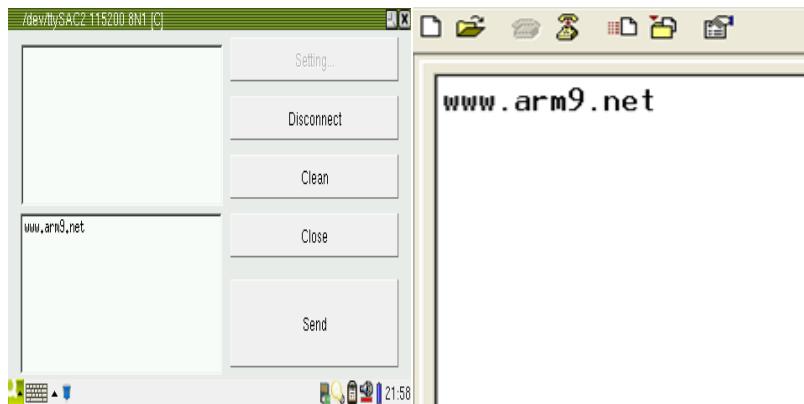
The title bar of the utility shows the default setting is “**ttySAC1 115200 8N1 [C]**”, and it implies the default port is:

- Serial Port Device: /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:

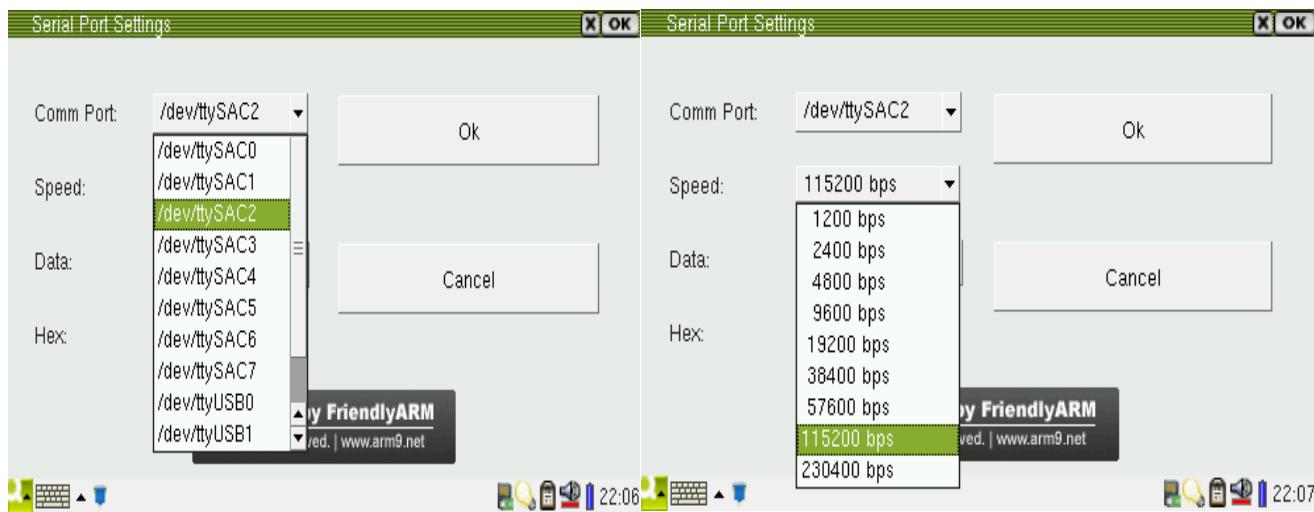
Comm Port: you can choose “/dev/ttySAC0,1,2” or the USB to Serial “/dev/ttyUSB0,1,2,3”

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



4.1.20 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 Mini6410-1405'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 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.

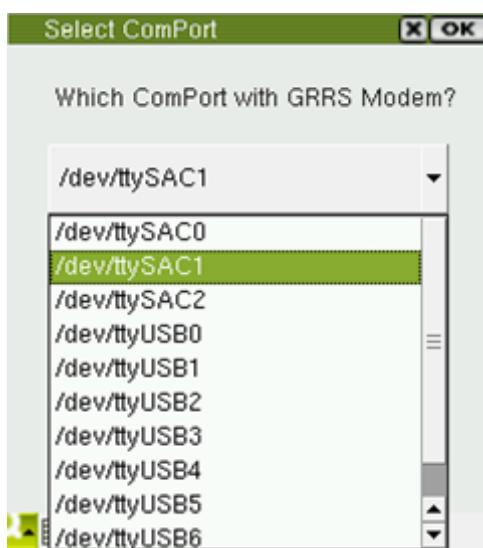
(2) Connect via USB to Serial cable

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

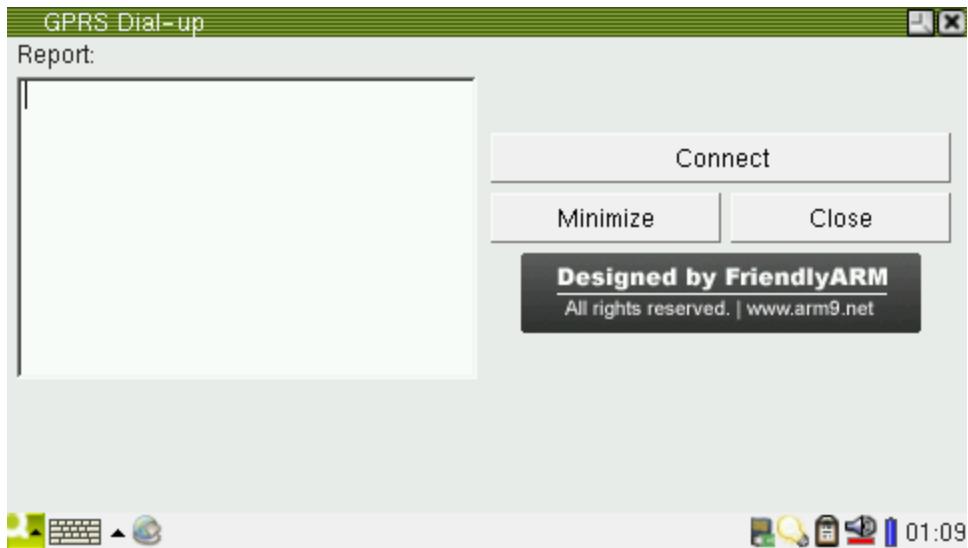


Note: after a USB to Serial device is inserted, you will find a new device “/dev/ttyUSB0” or “/dev/ttyUSB1” listed in the “/dev” directory

Go to the “Friendly ARM” 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.

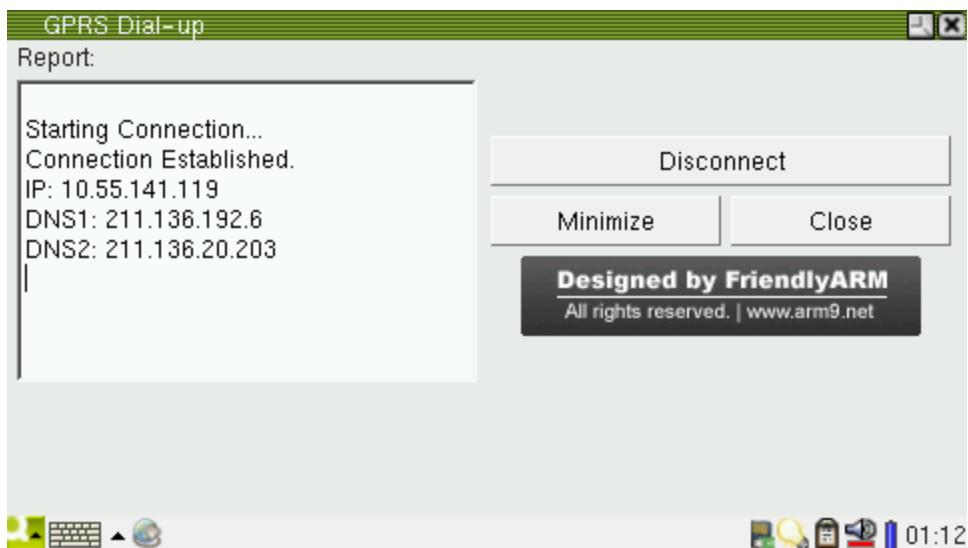


We select /dev/ttySAC1 and click on OK to continue.



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.



4.1.21 Messaging via GPRS Modem

Please go to “Friendly ARM” and click on “GPRS SMS”



If the connection is a success, the status will show “Ready”. Occasionally it shows “Device Init...” 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.

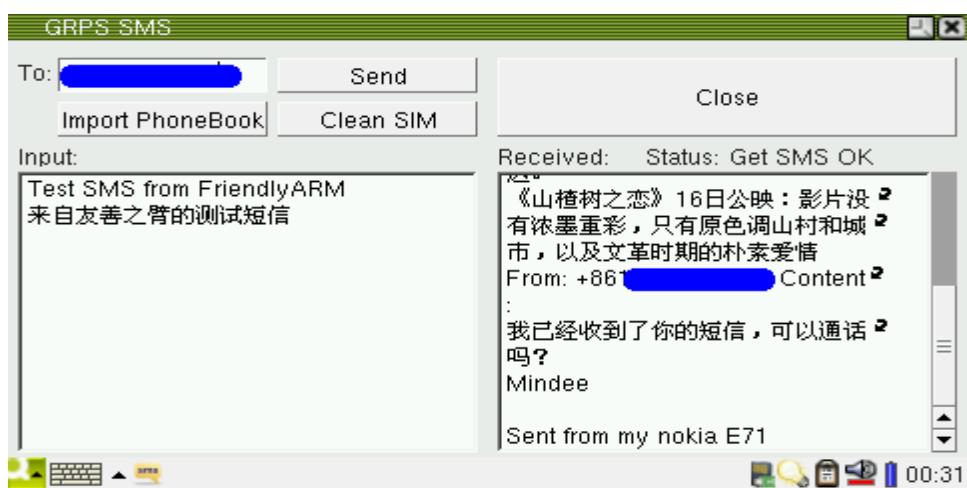
(1) Single sending messages

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.

4.1.22 3G Dial-Up

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



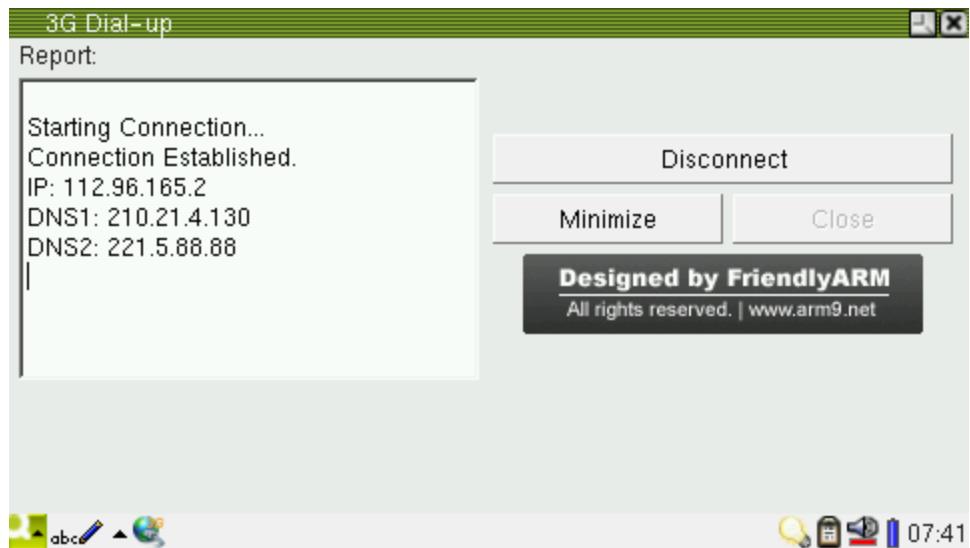
Step3: plug the network card into your board



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 Friendly ARM 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 Allegro 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 USBCConnect 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

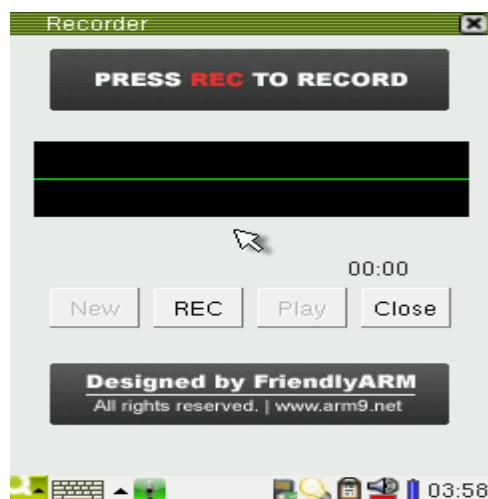
Vodafone MD950 (Wisue Technology)

Zydas ZD1211RWWLAN USB, Sphairon HomeLink 1202 (Variant 1)

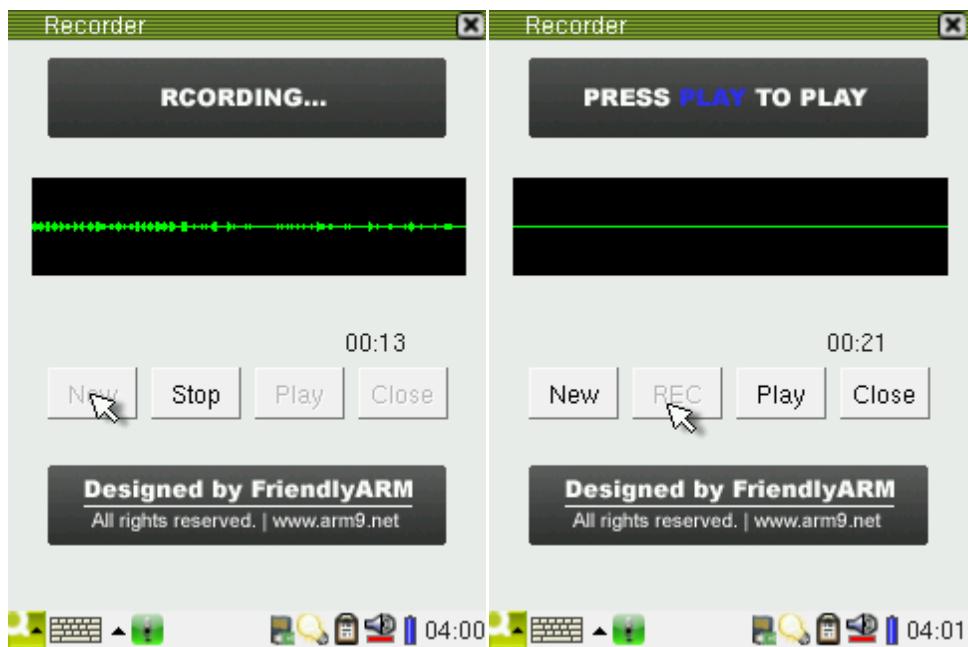
Zydas ZD1211RW WLAN USB, Sphairon HomeLink 1202 (Variant 2)

4.1.23 Audio Recording

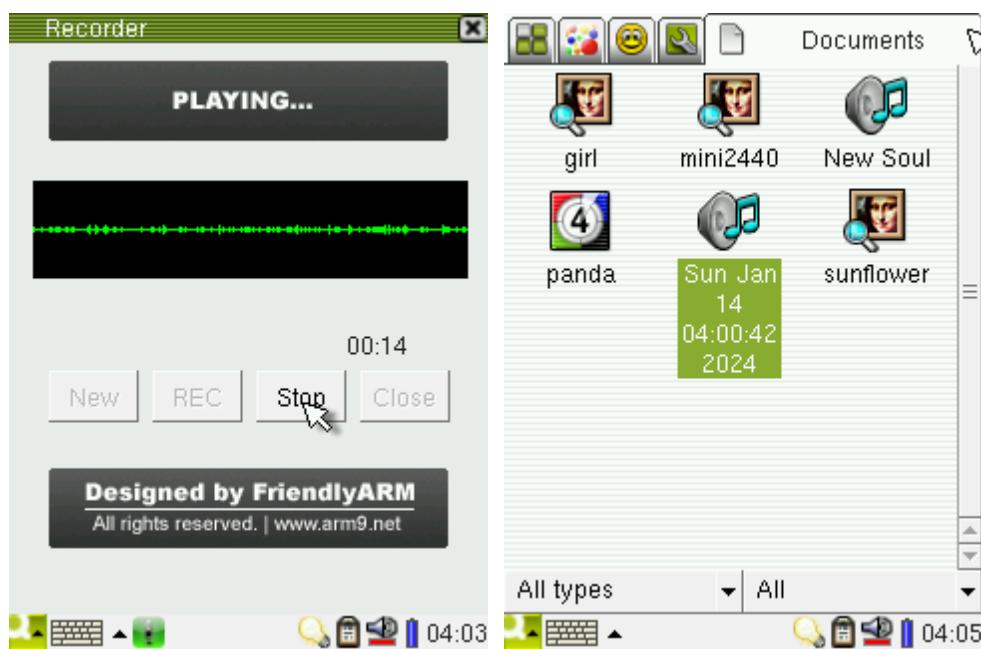
Go to the “FriendlyARM” tab and click on the “recorder” icon:



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.

4.1.24 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 email us capbily@163.com



4.1.25 Preview with Camera

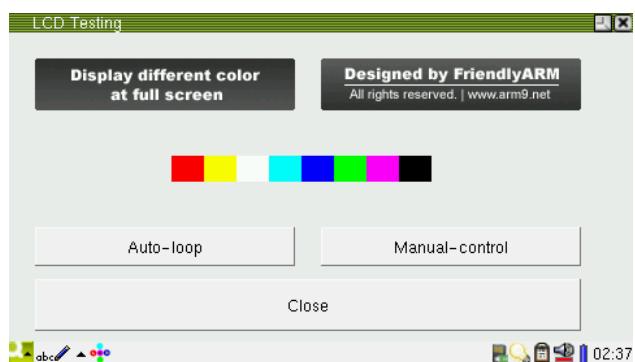
To launch the preview utility you need to use our shipped CMOS CAM130 module which also works with the MINI2440 system. Connect the module to your board, power on, go to the “Friendly ARM” tab and click on “CMOS Camera”. Clicking on “Snap” takes pictures of

what you 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.



4.1.26 LCD Test

Go to “FriendlyARM”, click on the “LED” icon you will see the following window:



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

4.1.27 Backlight Control

Note: this feature requires an LCD driven by the accurate touch driver.

If you already played our Mini6410-1405 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



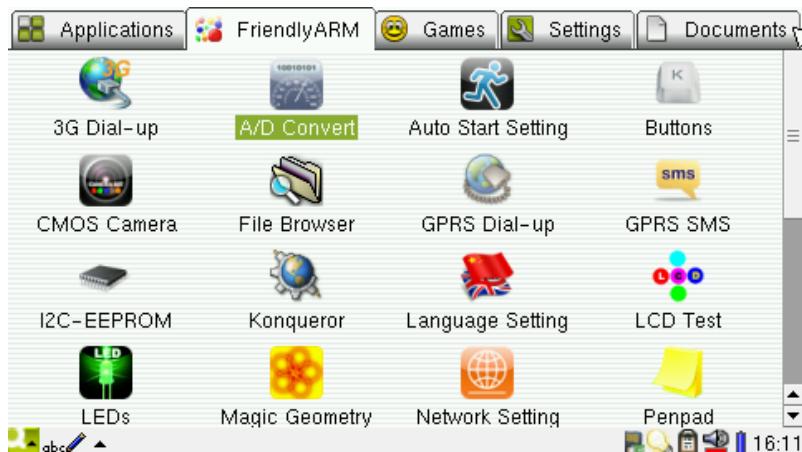
Here the default setting is 25 seconds you can click on the “Up” or “down” arrow to adjust it. If you uncheck “Light off”, the backlight will be on as long as the system is powered on.

An LCD driven by the accurate 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 4.2.1 for more details on how to adjust it via the command line utility.

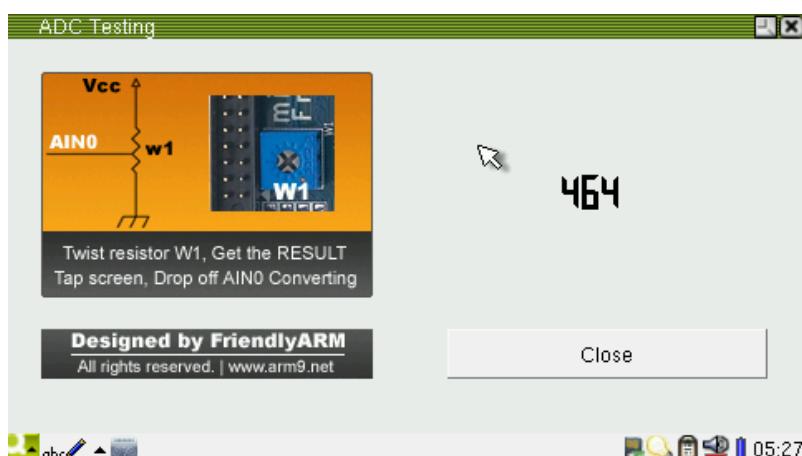
4.1.28 A/D Conversion

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:

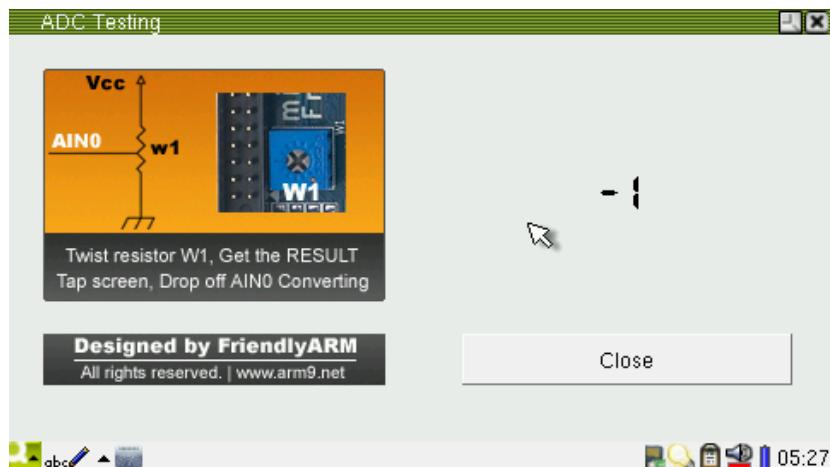
Click on the “ADC Testing” icon in the “FriendlyARM” tab:



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.

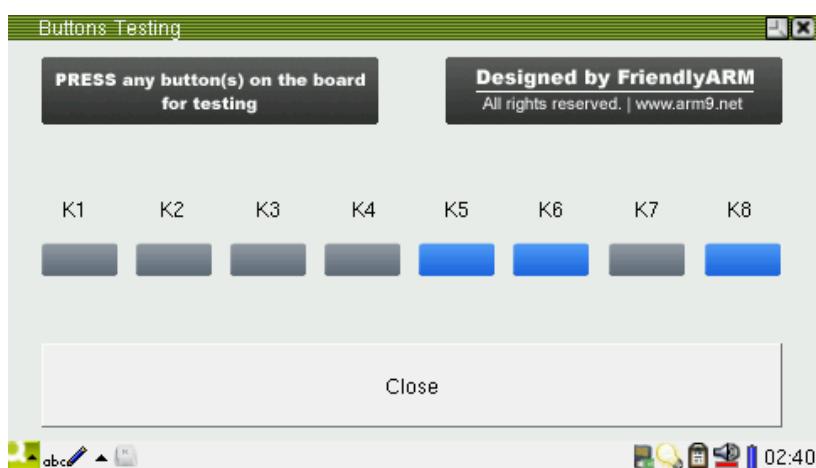


When you click on the touch screen, the A/D converter will take the touch screen as the channel, you will see the result “-1”; when you move your touch pen away from the screen, the A/D converter will take AIN0 as the channel again.



4.1.29 User Button Test

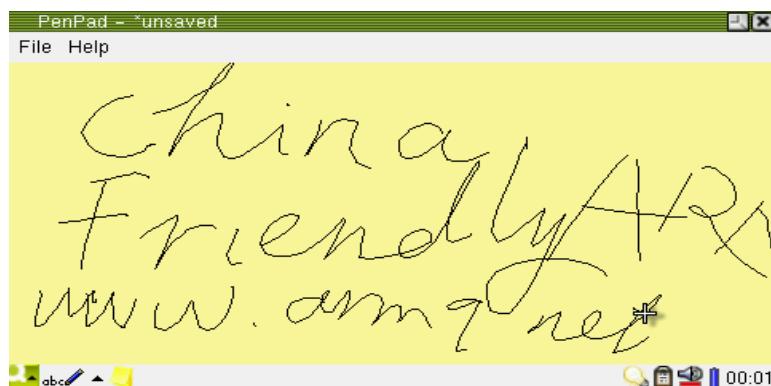
Note: the user buttons don't have dedicated functions and they are just for testing low level drivers. Click on the “Buttons” icon in the “FriendlyARM” tab. Press down any buttons on the board, the corresponding button icons will change to blue, release them, their icons will change back to grey.



4.1.30 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.

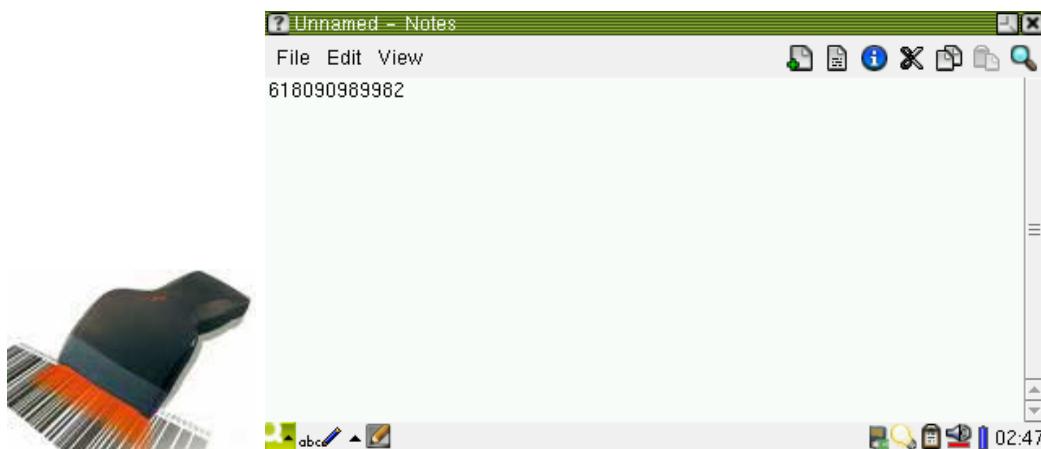


4.1.31 Barcode Scanner

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



4.1.32 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.



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).

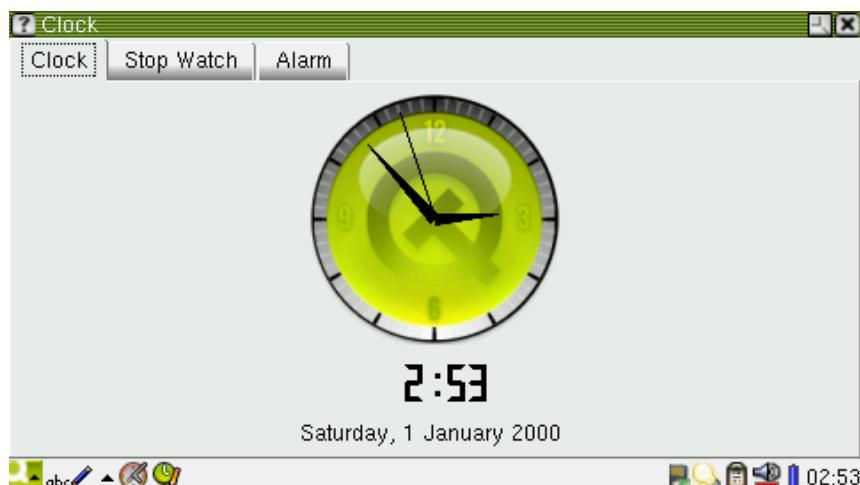


4.1.33 Set Up Time Zone, Date, Time and Alarm Clock

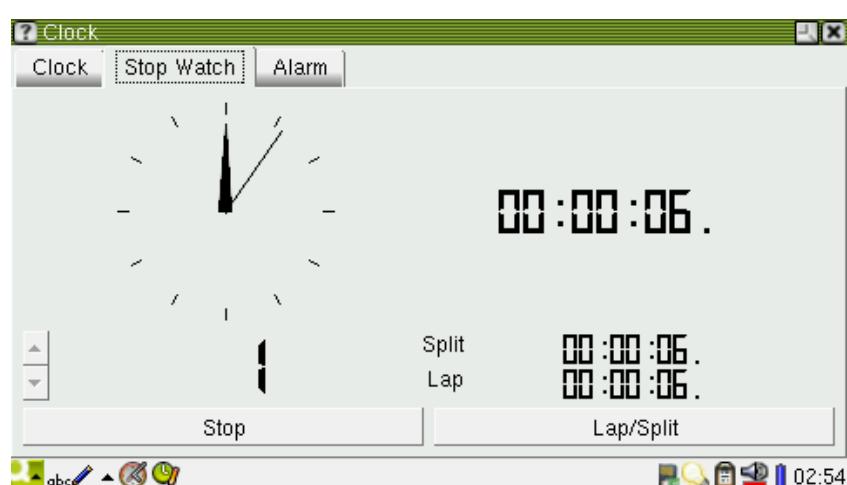
When you get our system, the date and time usually might not be accurate. You can adjust them by yourself. Because the CPU has its own RTC and the board has a backup battery, after you adjust the date and time, they will be saved. To adjust them, click on the time zone area at the right bottom of the 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.



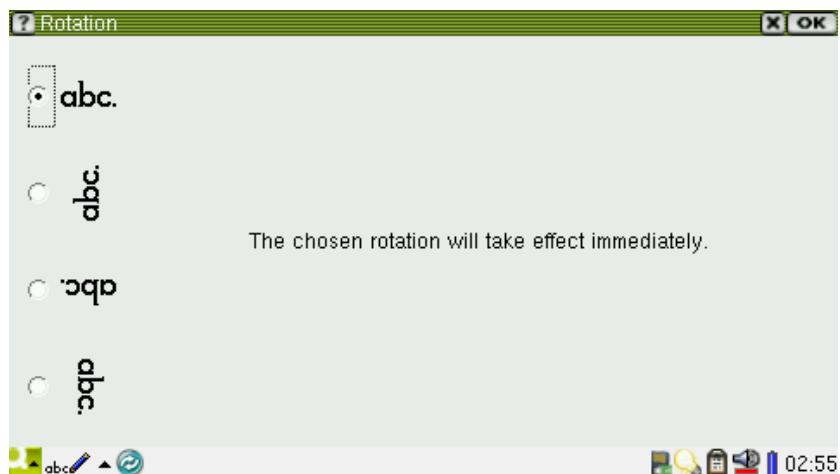
Click on “Stop Watch” to open a stopwatch utility



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.

4.1.34 Screen Rotation

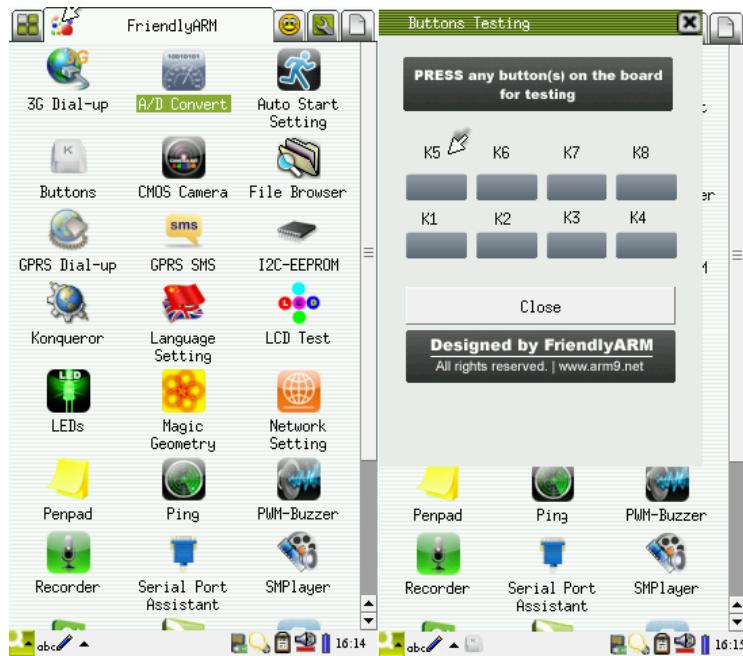
Click on the “rotation” icon in the “settings” tab to enter its interface. You can rotate the screen in four directions.



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.

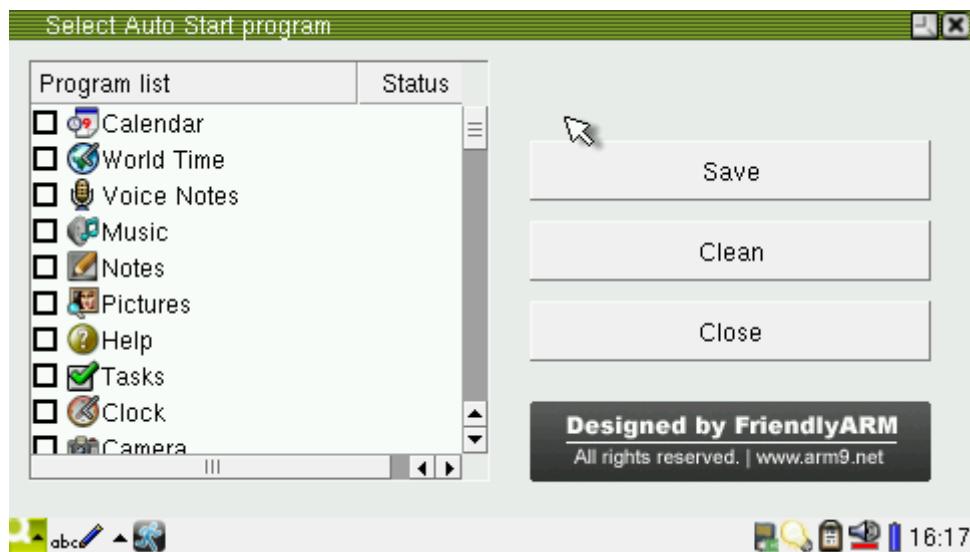
After rotation you will notice that all “Friendly ARM” utilities get rotated too. We implemented this feature to make our utilities displayed properly with different LCDs



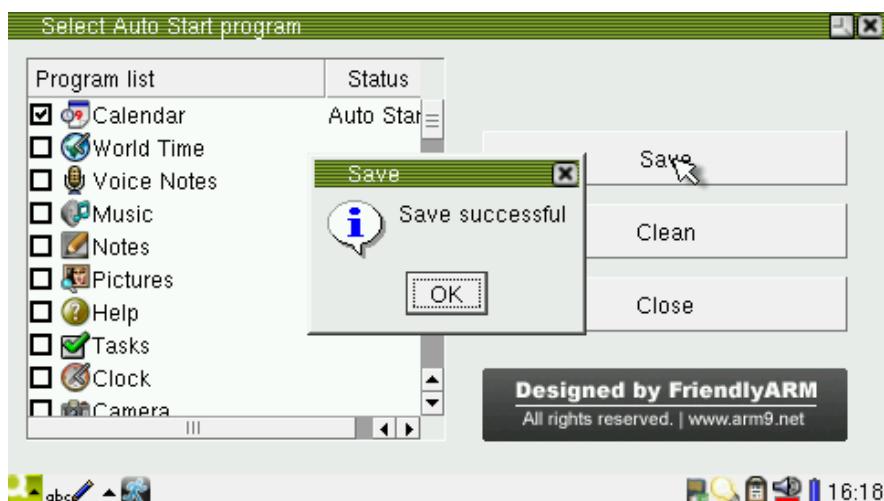
4.1.35 Setup Auto Run Program

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

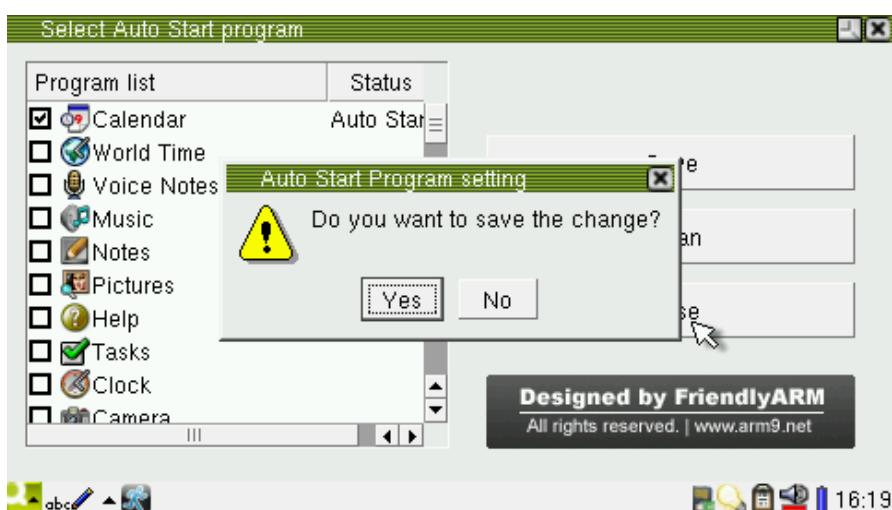
Click on the “Auto Start Setting” icon in the “FriendlyARM” tab.



Those 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.



4.1.36 System Shutdown

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

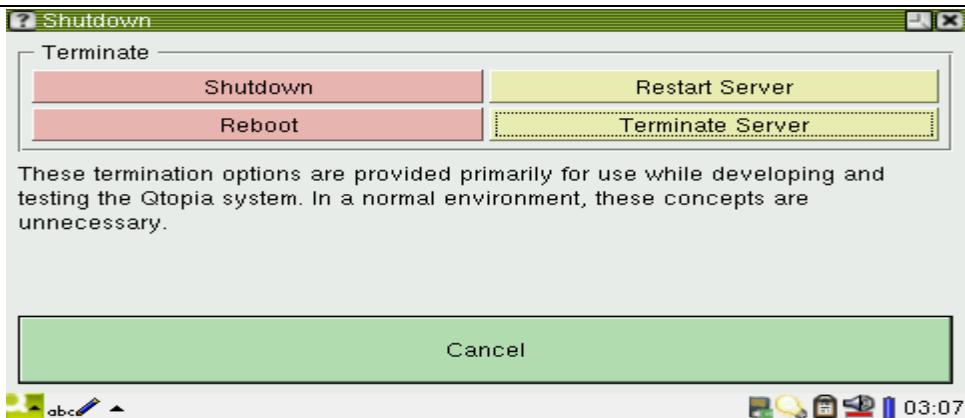
Shutdown: Press this button, Linux will end all the programs and services to shutdown the whole system. After the whole system is shutdown, the CPU will not be running and the system consumes less power. However since our system doesn’t have a hardware power down circuit you still can see the power LED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



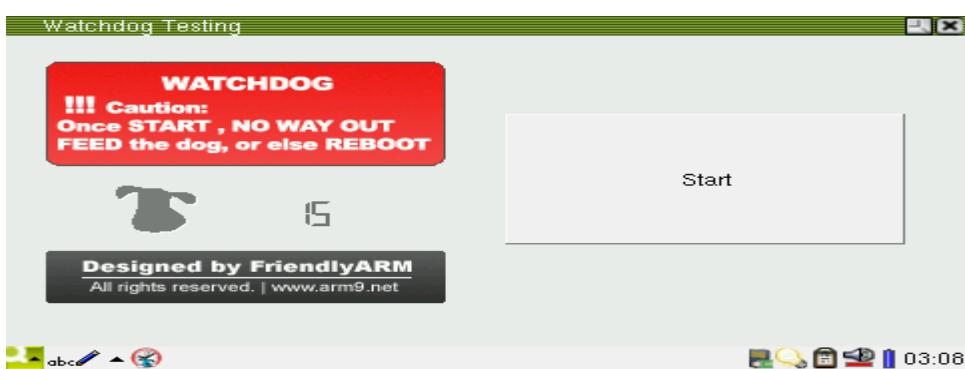
There is a “brightness and power” icon in the “setting” subgroup. Since our system doesn’t have a power management circuit, this icon is not active.

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

4.1.37 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.

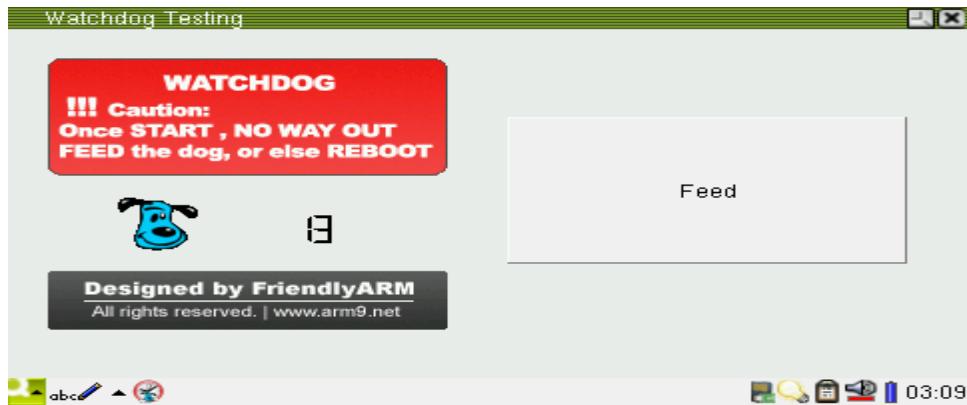
Click on the “Watchdog” icon in the “FriendlyARM” tab



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

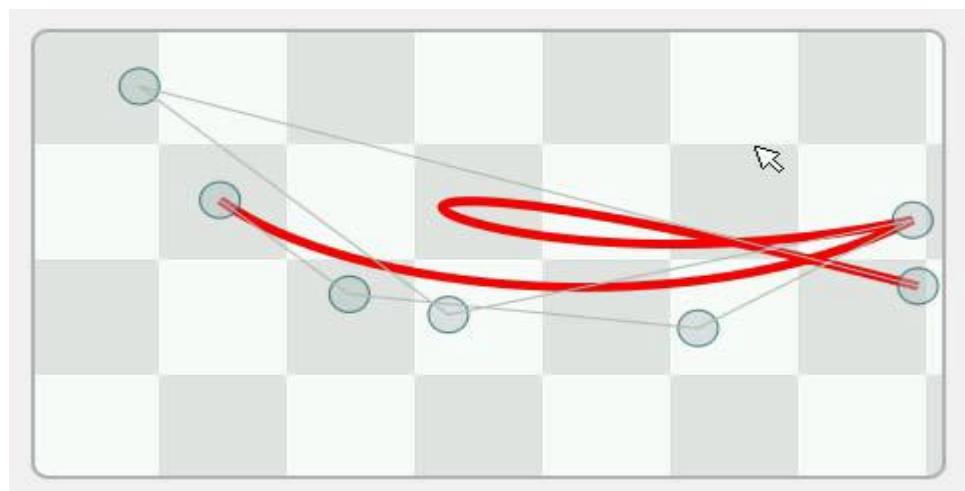
Here we set a countdown time 15 seconds. To feed the dog, click on the “Feed” button.

Keep feeding, it will always have bones and the system will not reboot.

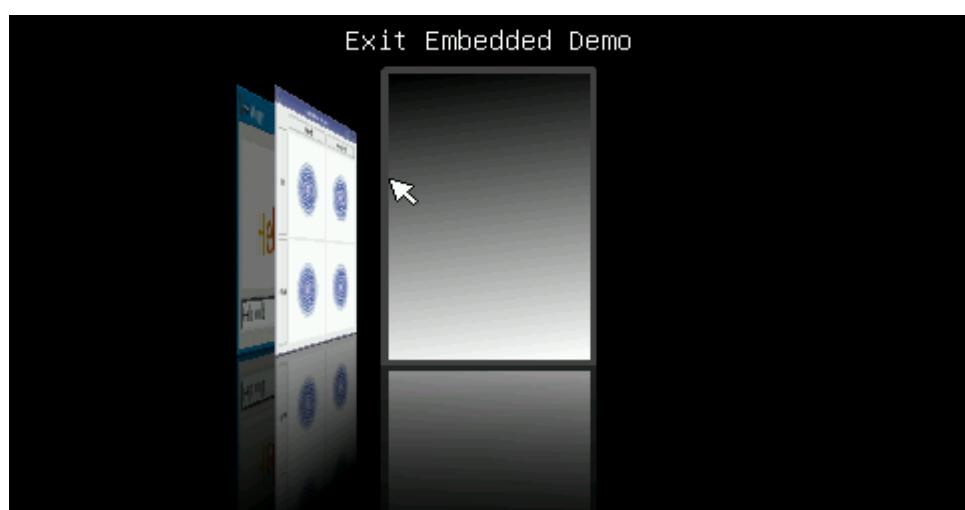


4.1.37 Start QtE-4.8.5

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



4.1.39 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-4.8.5 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-4.8.5 or high versions since they can work in more platforms and are easier for beginners to learn and migrate. In addition a QtE-4.8.5 application doesn't take too much memory.

4.2 Play Mini6410-1405 via Hyper Terminal

Note: every Linux fan may need to get familiar with the command line utility. All Linux commands are very similar 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.



```
s3c2410-rtc s3c2410-rtc: setting system clock to 2000-01-01 19:32:45 UTC (946755
165)
UBIFS: recovery needed
UBIFS: recovery completed
UBIFS: mounted UBI device 0, volume 0, name "FriendlyARM-root"
UBIFS: file system size: 253274112 bytes (247338 KiB, 241 MiB, 1963 LEBS)
UBIFS: journal size: 9033728 bytes (8822 KiB, 8 MiB, 71 LEBS)
UBIFS: media format: 4 (latest is 4)
UBIFS: default compressor: LZO
UBIFS: reserved for root: 0 bytes (0 KiB)
VFS: Mounted root (ubifs filesystem).
Freeing init memory: 124K
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will b
e case sensitive!
[01/Jan/2000:11:32:48 +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
-
```

4.2.1 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 Friendly ARM. 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 be 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 Linux-2.6.36, 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.

4.2.2 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.

4.2.3 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: O20070925A001033
usb 1-1: configuration #1 chosen from 1 choice
scsi0 : SCSI emulation for USB Mass Storage devices
scsi 0:0:0:0: Direct-Access      Hisun      Flash Disk      2.10 PQ: 0 ANSI: 2
sd 0:0:0:0: [sda] 4124664 512-byte hardware sectors: (2.11 GB/1.96 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] 4124664 512-byte hardware sectors: (2.11 GB/1.96 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
  sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
[root@FriendlyARM ~]# _
```

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

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

```
[root@FriendlyARM ~]# ls
I'm So Paid.mp3      Recycled          infinity 2008.mp3
[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/udisk on /udisk type vfat (rw,sync,nosuid,nodev,noatime,nodiratime,fmask=
22,dmask=0022,codepage=cp437,iocharset=iso8859-1)
[root@FriendlyARM ~]# _
```

4.2.4 Mount SD Card

Similar to USB drive mounting, a SD card will be automatically detected and mounted.

After inserting a 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: 0kHz)
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,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-1)
[root@FriendlyARM /]# _
```

4.2.5 File Transfer 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 Mini6410-1405 system via a serial port, you can transfer files to and from a host PC by using command “rz” or “sz” as follows:

(1) Transferring 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:



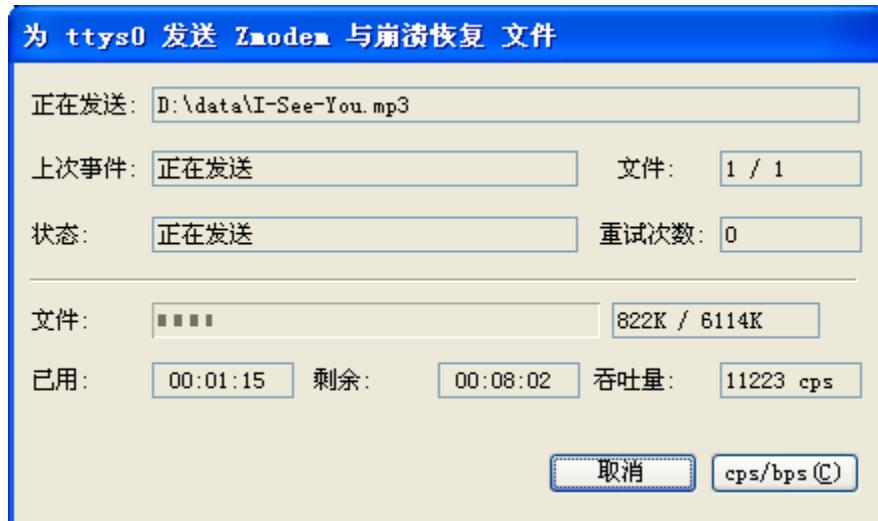
(2) Transferring files by using “rz”

In your Mini6410-1405 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.

4.2.6 LED Test

| led-player leds | | Comments |
|-----------------|---|----------|
| Source File | Led-player.c leds.c | |
| File Location | Linux\examples.tgz | |
| Device Name | /dev/leds | |
| Driver | Linux-2.6.36/drivers/char/mini6410_leds.c | |
| Notes: | | |

| Leds.cgi | | Comments |
|--|-----------------|-------------------|
| Source File | leds.cgi | In the shipped CD |
| File Location | See notes below | |
| Device Name | | |
| Driver | | |
| Notes: leds.cgi is a shell script. It is invoked by "leds.html". Uncompress "root_default.tgz" in the shipped CD you will find "leds.cgi" and "leds.html" in the www directory and you can open them in any text editor. | | |

Note: the led-player utility and the feature of controlling LEDs from an html page are all implemented by Friendly ARM originally for the SBC2410 system and can be easily migrated to other systems as well.

(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

4.2.7 User Button Test

| Buttons | | Comments |
|---------------|--|----------|
| Source File | Buttons_test.c | |
| File Location | See notes below | |
| Device Name | /dev/buttons | |
| Driver | Linux-2.6.36/drivers/char/mini6410_buttons.c | |

Notes: leds.cgi is a shell script. It is invoked by “leds.html”. Uncompress “root_default.tgz” in the shipped CD you will find “leds.cgi” and “leds.html” in the www directory and you can open them in any text editor.

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

```
[root@FriendlyARM /]# buttons
key 1 is down
key 1 is up
key 1 is down
key 2 is down
key 2 is up
key 1 is up
key 3 is down
key 3 is up
key 1 is down
key 1 is up
key 1 is down
key 5 is down
key 5 is up
key 1 is up
```

4.2.8 Serial Port Assistant

Note: the armcomtest utility is a straightforward and easy to use program developed by Friendly ARM 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

```

RPC: Registered udp transport module.
RPC: Registered tcp transport module.
s3c2410-rtc s3c2410-rtc: setting system clock to 2080-02-10 11:43:18 UTC (347479
0998)
yaffs: dev is 32505858 name is "mtdblock2"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.2, "mtdblock2"
yaffs_read_super: isCheckpointed 0
VFS: Mounted root (yaffs filesystem) on device 31:2.
Freeing init memory: 144K
hwclock: settimeofday() failed: Invalid argument
[05/Jan/1944:05:15:09 +0000] boa: server version Boa/0.94.13
[05/Jan/1944:05:15:09 +0000] boa: server built Feb 28 2004 at 21:47:23.
[05/Jan/1944:05:15:09 +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. eth0: link up, 100Mbps, full-duplex
, lpa 0x45E1

[root@FriendlyARM /]#
[root@FriendlyARM /]# armcomtest -d /dev/ttySAC1 -o
jjjjjjjjjxxxxxxxxxx_

```

4.2.9 PWM Buzzer

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.

```
[root@FriendlyARM ~]#
[root@FriendlyARM ~]#
[root@FriendlyARM ~]# pwm
pwd          pwm_test
[root@FriendlyARM ~]# pwm_test

BUZZER TEST ( PWM Control )
Press +/- to increase/reduce the frequency of BUZZER !
Press 'ESC' key to Exit this program !

      Freq = 1010
      Freq = 1020
      Freq = 1030
      Freq = 1020
      Freq = 1010
      Freq = 1000
```

4.2.10 Backlight Control

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

We also implement the feature of adjusting backlight for accurate 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.



4.2.11 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 /]#
```

4.2.12 AD Conversion

| adc-test | | Comments |
|---------------|--|----------|
| Source File | adc-test.c | |
| File Location | See notes below | |
| Device Name | /dev/adc | |
| Driver | Linux-2.6.36/drivers/char/mini6410_adc.c | |
| Notes: | | |

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

```
[root@FriendlyARM /]# adc-test
press Ctrl-C to stop
ADC Value: 0
ADC Value: 0
ADC Value: 0
ADC Value: 152
ADC Value: 295
ADC Value: 559
ADC Value: 800
ADC Value: 882
ADC Value: 890
ADC Value: 891
ADC Value: 892
```

4.2.13 USB WiFi/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 Mini6410-1405 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% FriendlyARM4(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:

```
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% 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 /]#
```

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 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 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 /]# _
```

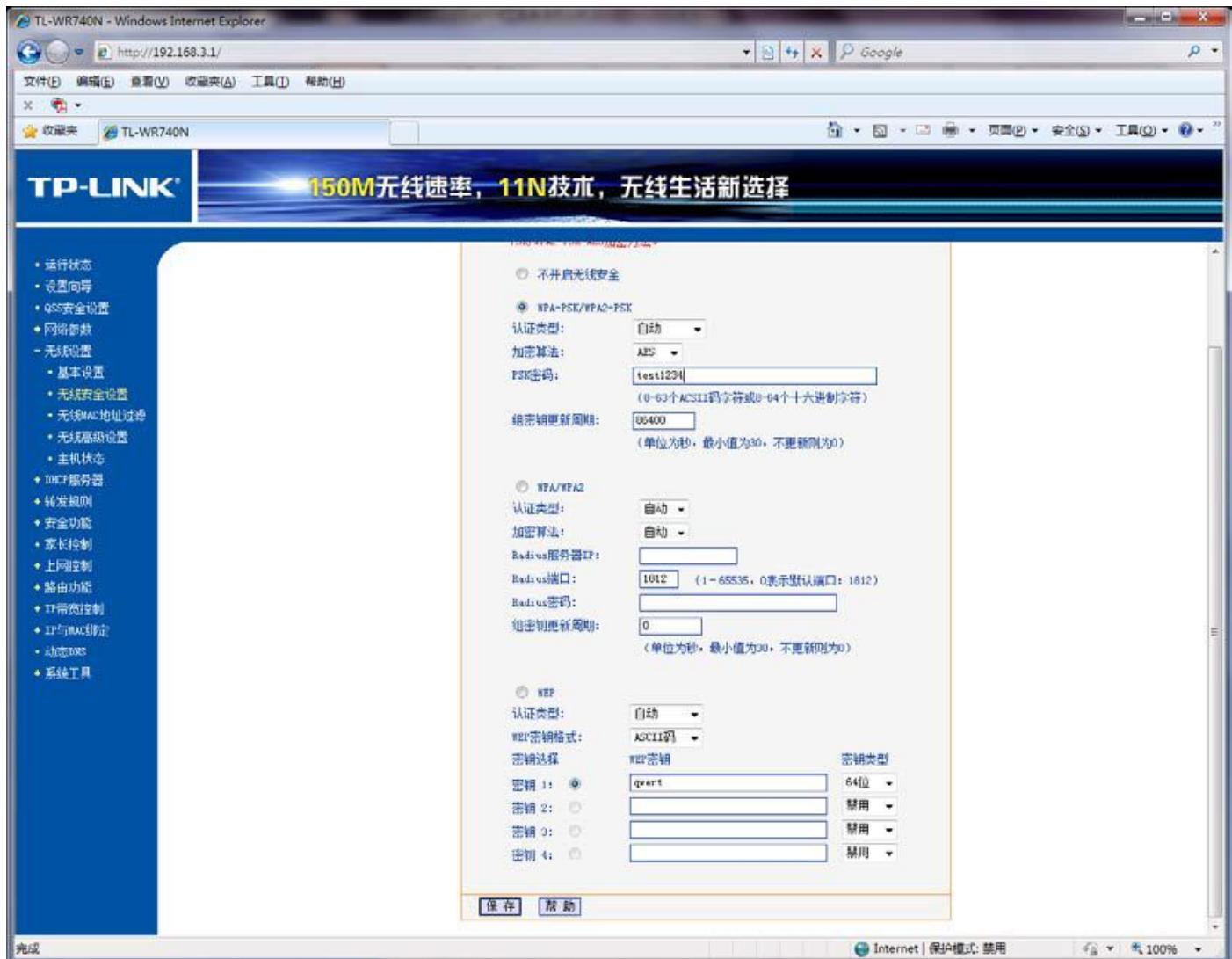
Or you can try this IP in your PC's web browser.



2.2 Connecting 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

```
[root@FriendlyARM /]# scan-wifi
    74% FriendlyARM4(Security)
    100% FriendlyARM-Test(Security)
2 Access Point Found
[root@FriendlyARM /]# _
```

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

```
[root@FriendlyARM /]# scan-wifi
    74% FriendlyARM4(Security)
    100% FriendlyARM-Test(Security)
2 Access Point Found
[root@FriendlyARM /]# start-wifi wpa FriendlyARM-Test test1234
sh: cannot kill pid 794: No such process
cfg80211: Calling CRDA to update world regulatory domain
udhcpc (v1.13.3) started
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 /]# scan-wifi
    74% FriendlyARM4(Security)
    100% FriendlyARM-Test(Security)
2 Access Point Found
[root@FriendlyARM /]# start-wifi wpa FriendlyARM-Test test1234
sh: cannot kill pid 794: No such process
cfg80211: Calling CRDA to update world regulatory domain
udhcpc (v1.13.3) started
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=31.870 ms
64 bytes from 192.168.3.1: seq=1 ttl=64 time=76.107 ms
64 bytes from 192.168.3.1: seq=4 ttl=64 time=42.121 ms
```

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



3. Closing USB WiFi Connection

Command “stop-wifi” to close your USB WiFi connection

4.2.14 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.

```

-sh: can't access tty; job control turned off
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:3E:26:0A:5B
          inet addr:192.168.1.230  Bcast:192.168.1.255  Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:14 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:1193 (1.1 KiB)  TX bytes:0 (0.0 B)
                  Interrupt:53 Base address:0x300

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

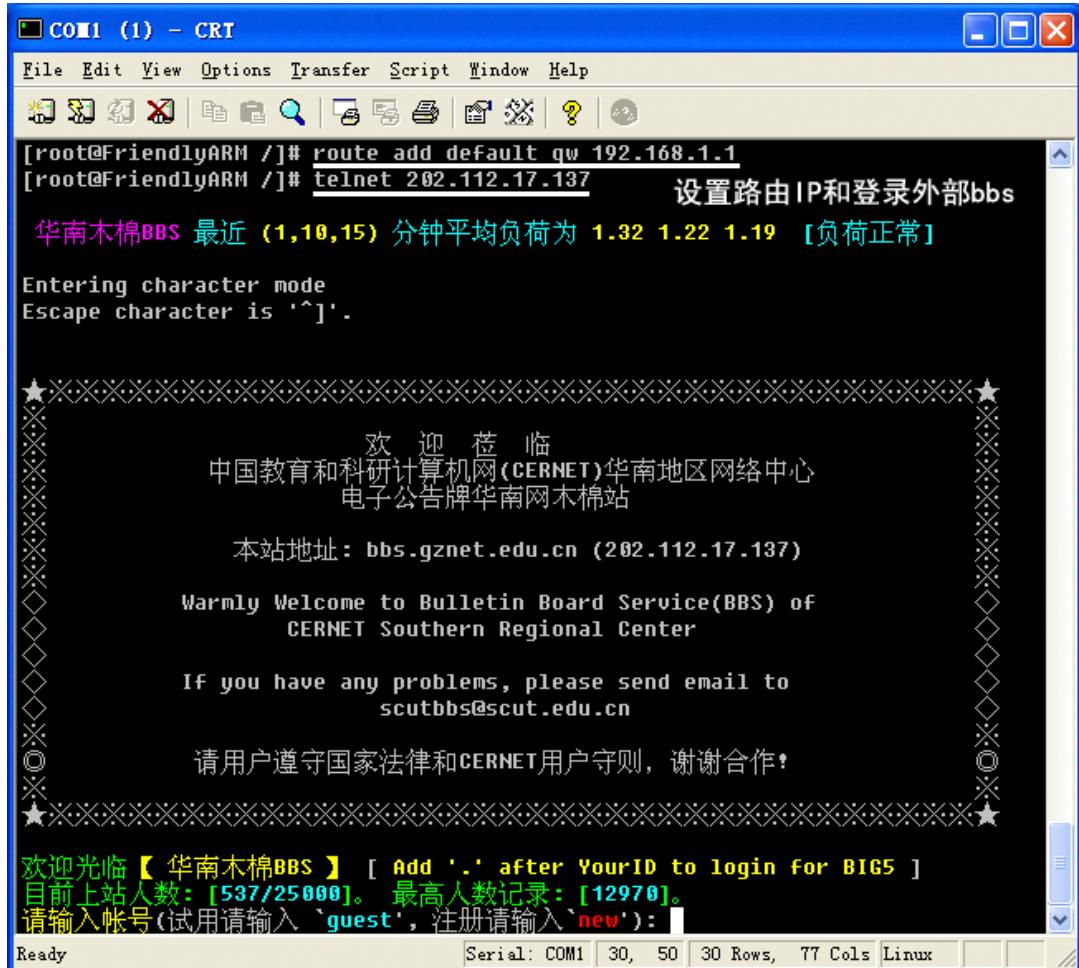
[root@FriendlyARM /]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=6.5 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.9 ms

```

Connection Successful

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”.



4.2.15 Ethernet Configuration

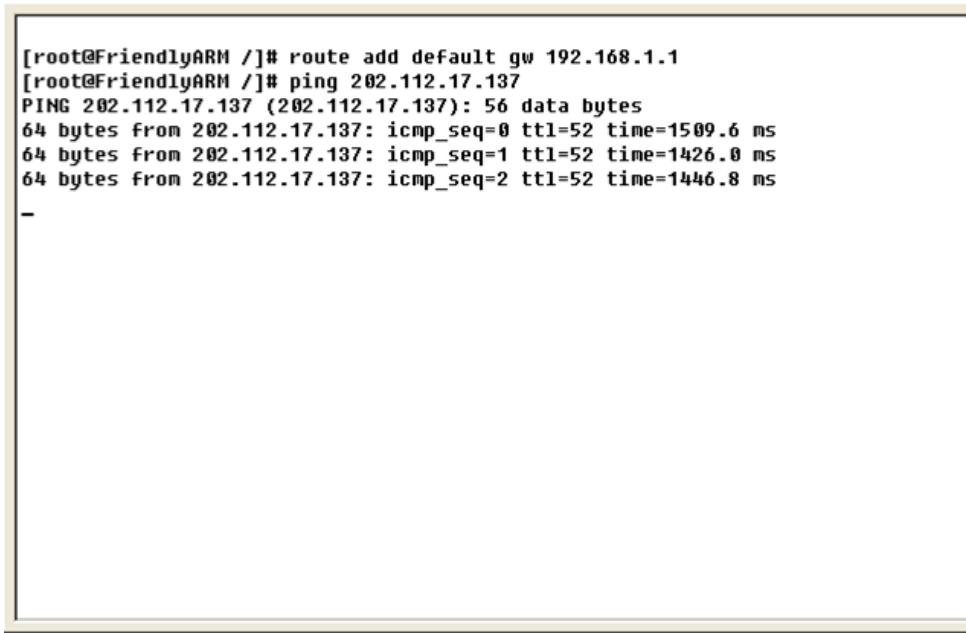
Connect your board to the internet, write down your gateway IP(the one in our example was 192.168.1.1) and configure your router:

```
# 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):

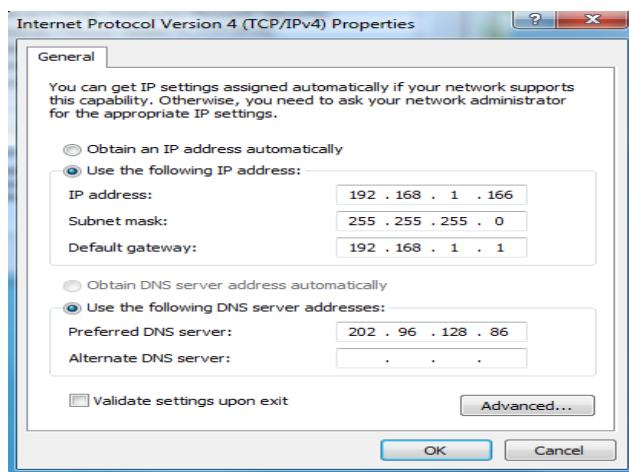
```
#ping 202.112.17.137
```

If it is a success you will see the following output



```
[root@FriendlyARM /]# route add default gw 192.168.1.1
[root@FriendlyARM /]# ping 202.112.17.137
PING 202.112.17.137 (202.112.17.137): 56 data bytes
64 bytes from 202.112.17.137: icmp_seq=0 ttl=52 time=1509.6 ms
64 bytes from 202.112.17.137: icmp_seq=1 ttl=52 time=1426.0 ms
64 bytes from 202.112.17.137: icmp_seq=2 ttl=52 time=1446.8 ms
```

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



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

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

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

#echo nameserver 202.96.128.86 >> /etc/resolv.conf; Set up the DNS configuration file

resolv.conf with your DNS IP or you can edit it with vi.

```
[root@FriendlyARM /]# rm /etc/resolv.conf
[root@FriendlyARM /]# touch /etc/resolv.conf
[root@FriendlyARM /]# echo nameserver 202.96.128.86 >> /etc/resolv.conf
[root@FriendlyARM /]# cat /etc/resolv.conf
nameserver 202.96.128.86
[root@FriendlyARM /]# ping www.163.com
PING www.cache.split.163.com (220.181.28.54): 56 data bytes
64 bytes from 220.181.28.54: icmp_seq=0 ttl=53 time=1353.8 ms
64 bytes from 220.181.28.54: icmp_seq=1 ttl=53 time=1378.0 ms
64 bytes from 220.181.28.54: icmp_seq=2 ttl=53 time=1398.1 ms
64 bytes from 220.181.28.54: icmp_seq=3 ttl=53 time=1356.0 ms
64 bytes from 220.181.28.54: icmp_seq=4 ttl=53 time=1314.9 ms
64 bytes from 220.181.28.54: icmp_seq=5 ttl=53 time=1314.9 ms
--- www.cache.split.163.com ping statistics ---
7 packets transmitted, 5 packets received, 28% packet loss
round-trip min/avg/max = 1314.9/1360.1/1398.1 ms
[root@FriendlyARM /]# _
```

4.2.16 Set MAC Address

The MAC address in the Mini6410-1405 is “soft” therefore you can change it via “ifconfig”.

First check your current MAC address via “ifconfig”:

#ifconfig ;

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

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
[root@FriendlyARM /]# _
```

In our example the MAC was “08: 90: 90: 90: 90: 90”, this is the default MAC address and has been hard-coded in the kernel. If you want to update it you have to recompile the

kernel. In order to change the MAC dynamically you need to close your network connection and then fill your new MAC:

```
#ifconfig eth0 down
```

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

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

```
#ifconfig eth0 up
```

```
#ifconfig
```

```
#ping 192.168.1.1
```

```
[root@FriendlyARM /]# ifconfig eth0 hw ether 00:11:aa:bb:cc:dd
[root@FriendlyARM /]# ifconfig eth0 up
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet HWaddr 00:11:AA:BB:CC:DD
          inet addr:192.168.1.230 Bcast:192.168.1.255 Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:68 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:4791 (4.6 KiB) TX bytes:672 (672.0 B)
                  Interrupt:53 Base address:0x300

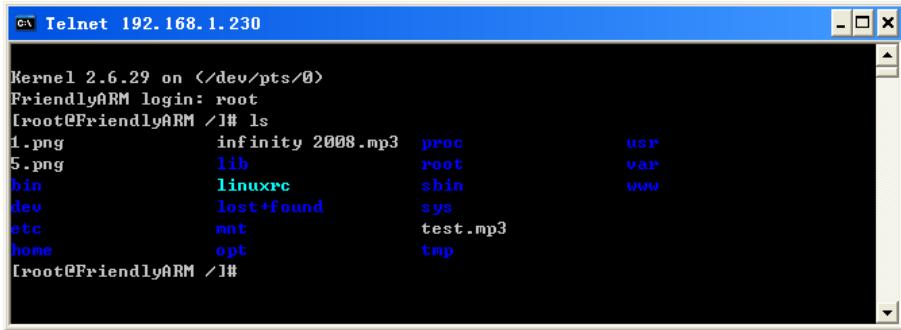
lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
                  UP LOOPBACK RUNNING MTU:16436 Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0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

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

4.2.17 Telnet Mini6410-1405

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.



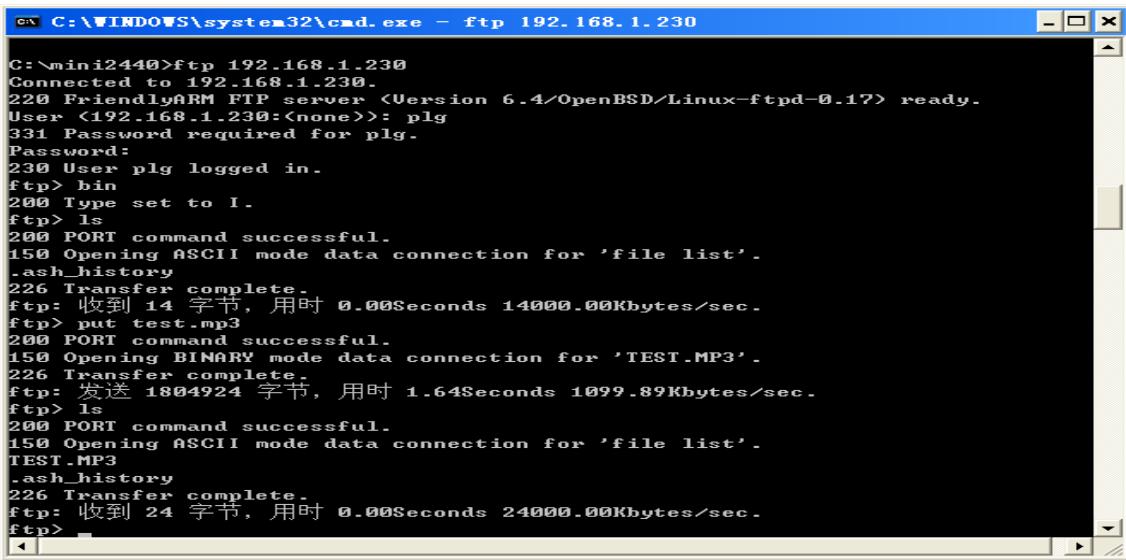
```
Kernel 2.6.29 on </dev/pts/0>
FriendlyARM login: root
[root@FriendlyARM ~]# ls
1.png      infinity 2008.mp3  proc      usr
5.png      lib       root      var
bin        linuxrc   shin      www
dev        lost+found sys
etc        mnt      test.mp3
home      opt      tmp
[root@FriendlyARM ~]#
```

4.2.18 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 done you will see a test.mp3 file in your board’s /home/plg directory.



```
C:\> C:\WINDOWS\system32\cmd.exe - ftp 192.168.1.230
C:\> miniz2440>ftp 192.168.1.230
Connected to 192.168.1.230.
220 FriendlyARM FTP server (Version 6.4/OpenBSD/Linux-ftp-0.17) ready.
User <192.168.1.230:<none>>: plg
331 Password required for plg.
Password:
230 User plg logged in.
ftp> bin
200 Type set to I.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for 'file list'.
..ash_history
226 Transfer complete.
ftp: 收到 14 字节, 用时 0.00Seconds 14000.00Kbytes/sec.
ftp> put test.mp3
200 PORT command successful.
150 Opening BINARY mode data connection for 'TEST.MP3'.
226 Transfer complete.
ftp: 发送 1804924 字节, 用时 1.64Seconds 1099.89Kbytes/sec.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for 'file list'.
TEST.MP3
..ash_history
226 Transfer complete.
ftp: 收到 24 字节, 用时 0.00Seconds 24000.00Kbytes/sec.
ftp>
```

4.2.19 Manipulate LED via WEB

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
```

4.2.20 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 noblock 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
```

```
[root@FriendlyARM /]# mount -t nfs -o noblock 192.168.1.111:/opt/FriendlyARM/QQ240/root_nfs /mnt      mount NFS to /mnt
[root@FriendlyARM /]# ls /mnt/
bin          lib          proc          usr
dev          linuxrc      sbin          var
etc          mnt          shanghaiwan.mp3  www
home         opt          tmp
[root@FriendlyARM /]# cd /mnt/
[root@FriendlyARM /mnt]# madplay shanghaiwan.mp3
MPEG Audio Decoder 0.15.0 (beta) - Copyright (c) 2000-2003 Robert Leslie et al.
    Title: 上海滩
    Artist: 叶丽仪
    Year: 2000
    Genre: Goa
                                Play MP3 in NFS
```

4.2.21 Set 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.

4.2.22 Save Data to Nand Flash

The Mini6410-1405 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 / shanghai.mp3 /home/plg
```

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

4.2.23 Set Up Auto Run Program

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/alsactl 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
```

4.2.24 Take Screenshots with Snapshot

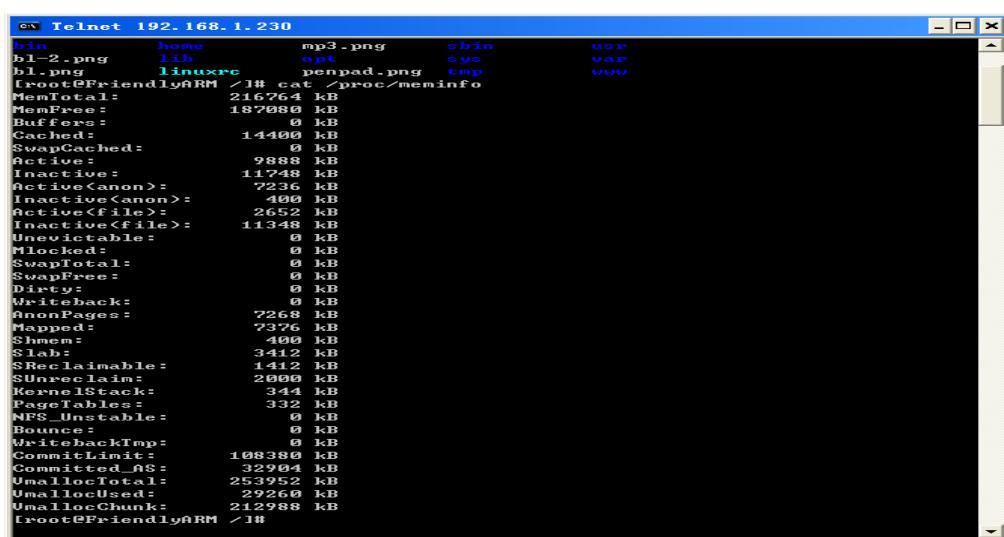
Users can take screenshots with “snapshot” and save them as png files

```
#snapshot pic.png
```

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

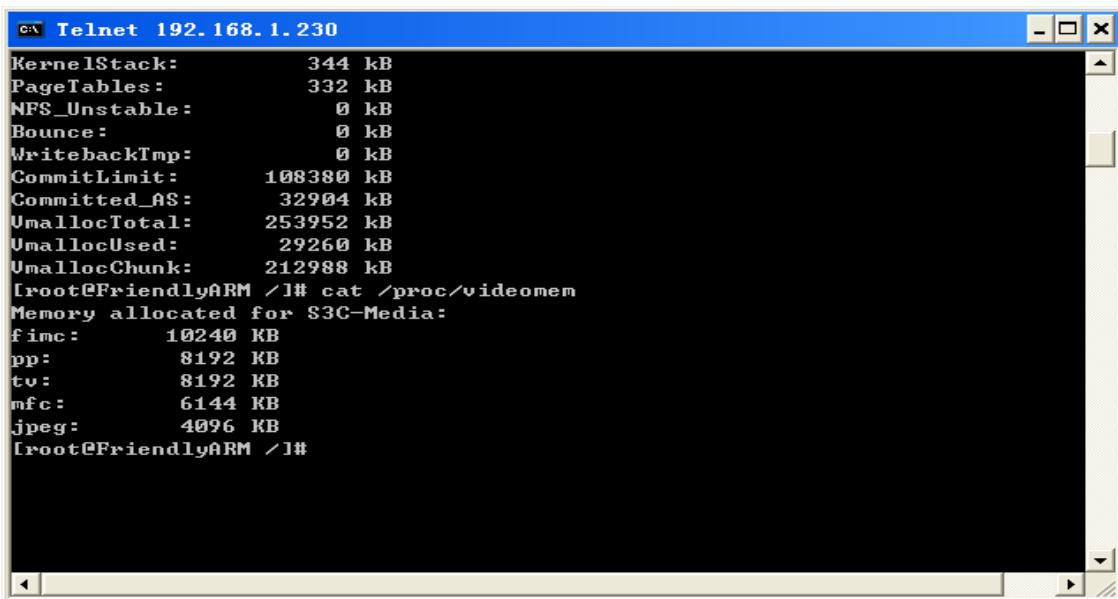
4.2.25 Check RAM Info

The Mini6410-1405 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



```
bin          home      mp3.png      skin      user
bl-2.png    lib        opt         sos       var
bl.png      linuxrc   penpad.png  tmp       www
[root@FriendlyARM ~]# cat /proc/meminfo
MemTotal:       216764 kB
MemFree:        187080 kB
Buffers:        0 kB
Cached:         14400 kB
Writeback:      0 kB
Inactive:       9888 kB
Inactive:       11248 kB
Active(anon):   2236 kB
Inactive(anon): 400 kB
Active(file):   2652 kB
Inactive(file): 11348 kB
Unevictable:    0 kB
Mlocked:        0 kB
SwapTotal:      0 kB
SwapFree:       0 kB
Dirty:          0 kB
Writeback:      0 kB
AnonPages:      2268 kB
Mapped:         7376 kB
Shmem:          400 kB
Slab:           3412 kB
SReclaimable:   1412 kB
SUnreclaim:     2000 kB
KernelStack:    344 kB
PageTables:     332 kB
NFS_Unstable:   0 kB
Bounce:          0 kB
WritebackTmp:   0 kB
Commitlimit:    109380 kB
Committed_AS:  32904 kB
UmallocTotal:   253952 kB
UmallocUsed:   29260 kB
UmallocChunk:  212988 kB
[root@FriendlyARM ~]#
```

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

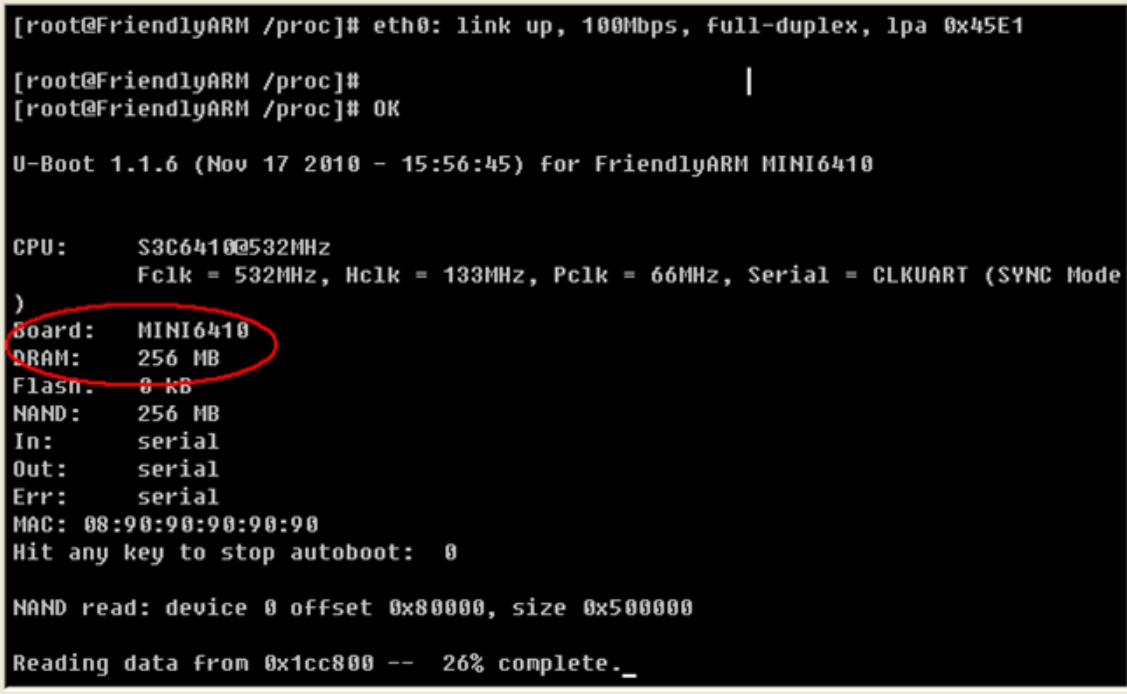


```

Telnet 192.168.1.230
KernelStack:      344 kB
PageTables:       332 kB
NFS_Unstable:     0 kB
Bounce:           0 kB
WritebackTmp:     0 kB
CommitLimit:     108380 kB
Committed_AS:    32904 kB
UmallocTotal:    253952 kB
UmallocUsed:     29260 kB
UmallocChunk:    212988 kB
[root@FriendlyARM ~]# cat /proc/videomem
Memory allocated for S3C-Media:
fmc:        10240 KB
pp:         8192 KB
tv:         8192 KB
mfc:        6144 KB
jpeg:       4096 KB
[root@FriendlyARM ~]#

```

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:   8 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.-

```

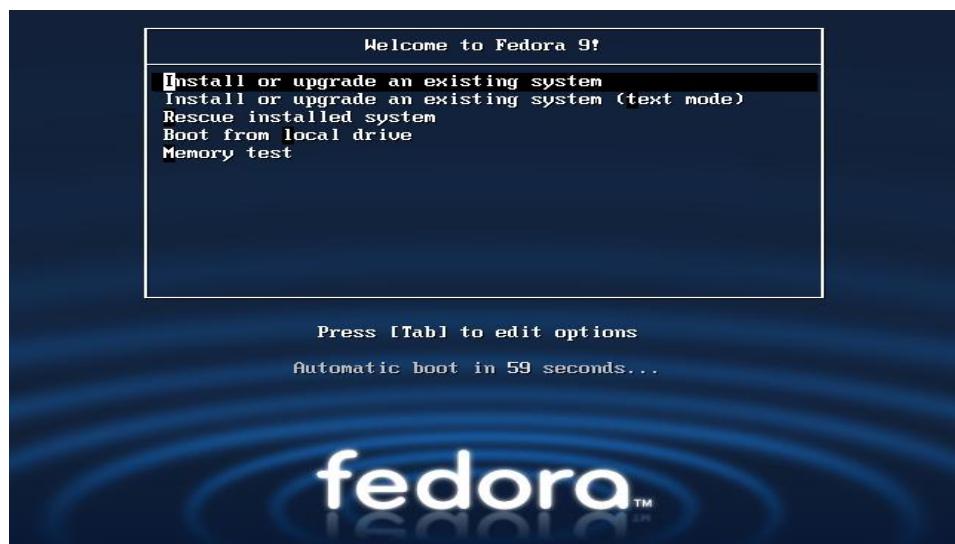
4.3 Set up Fedora 9.0 Development Environment

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 Mini6410-1405 were based on Fedora 9.0. We didn't do it on other platforms. We strongly suggest that our users should use this platform 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>).

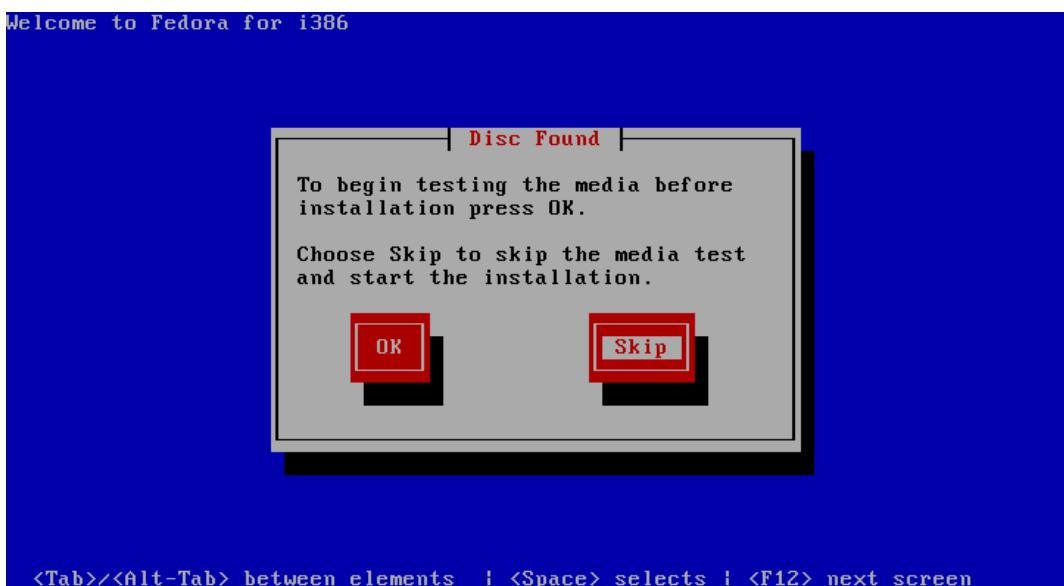
The reason why we chose Fedora 9.0 is that it is easy to be installed and set up. Fedora 10 and later versions are more complicated and therefore may not be easy for beginners and Fedora 8 and earlier versions are a little bit obsolete. Please follow the steps below to install.

4.3.1 Install Fedora 9.0

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



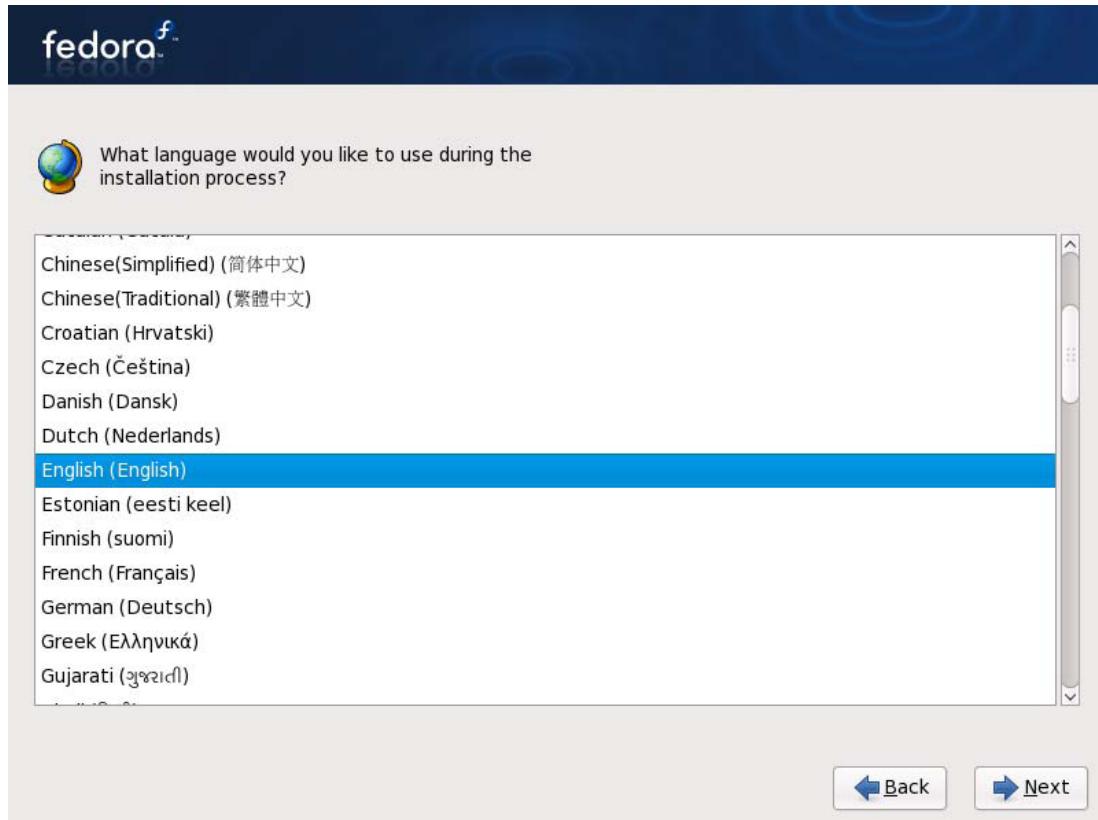
Step2: The system will check the installation disk. It can be ignored, just press “Skip” to the next step



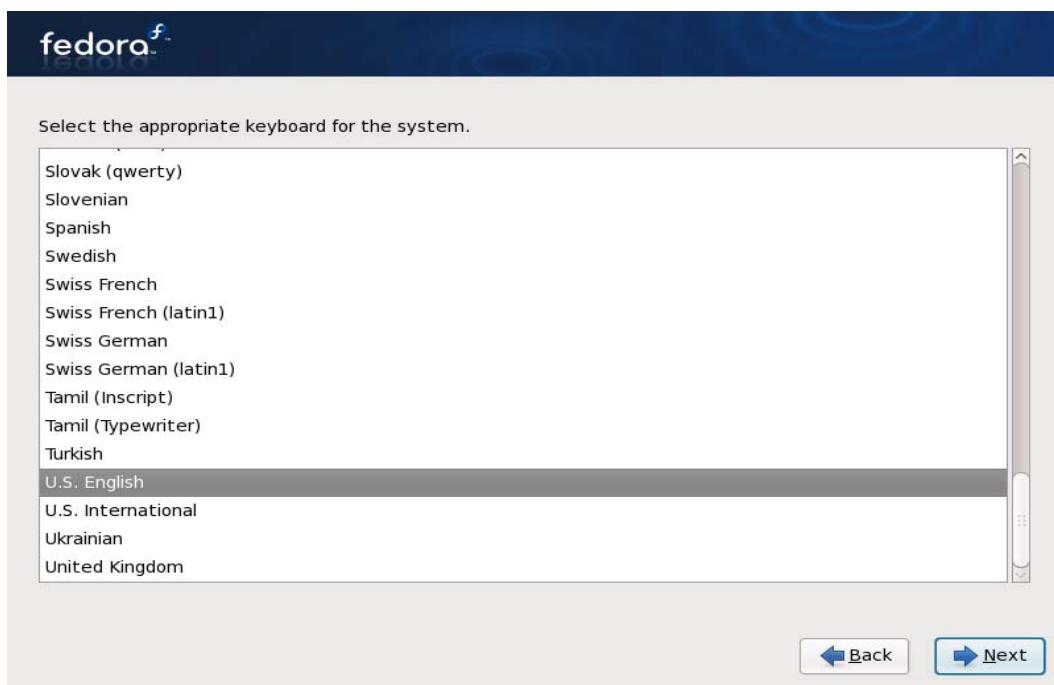
Step3: it enters the graphic interface, click on the “Next” button.



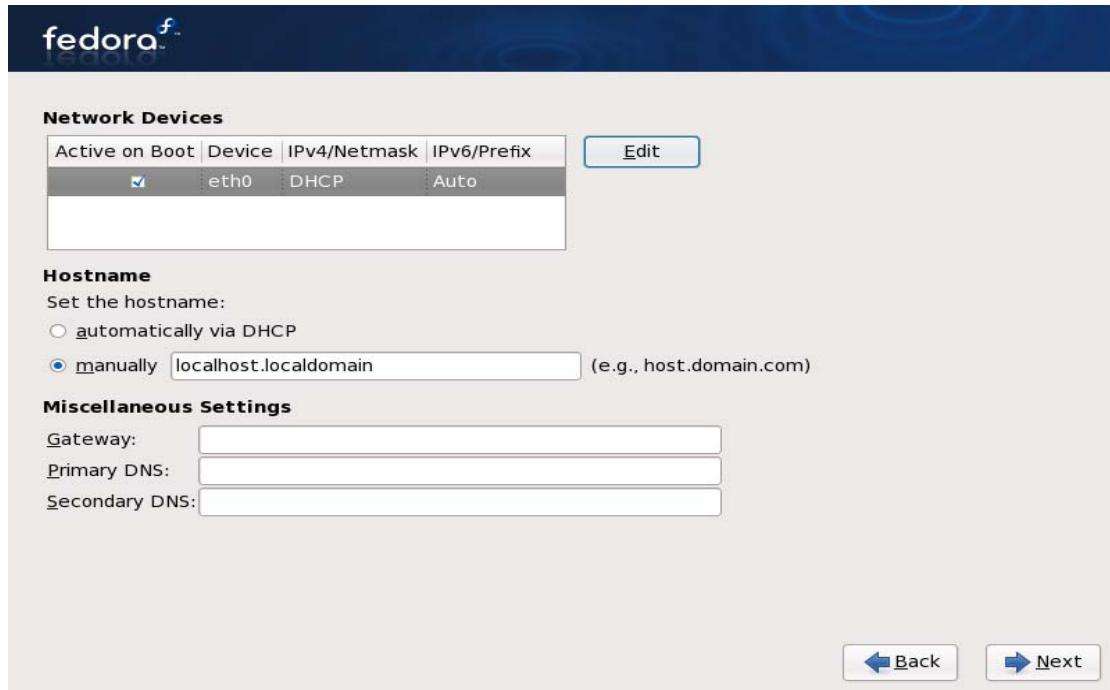
Step4: set the installation language. In this example, we chose the simplified English.



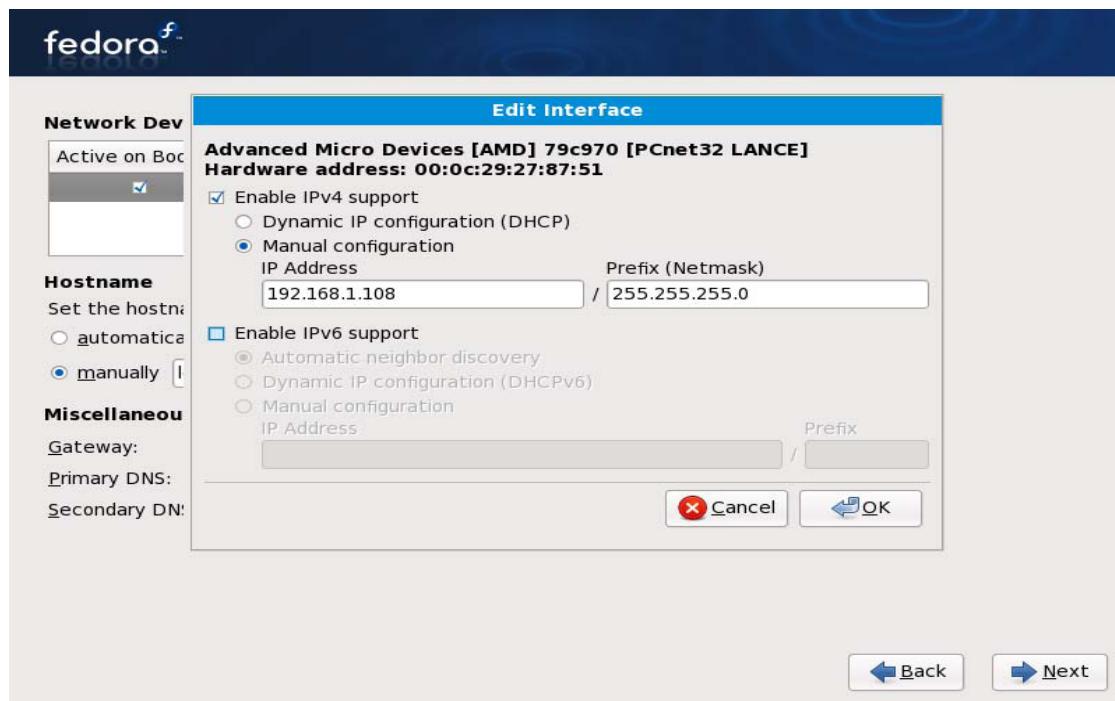
Step5: set the keyboard, in this example, we chose the U.S. key board.



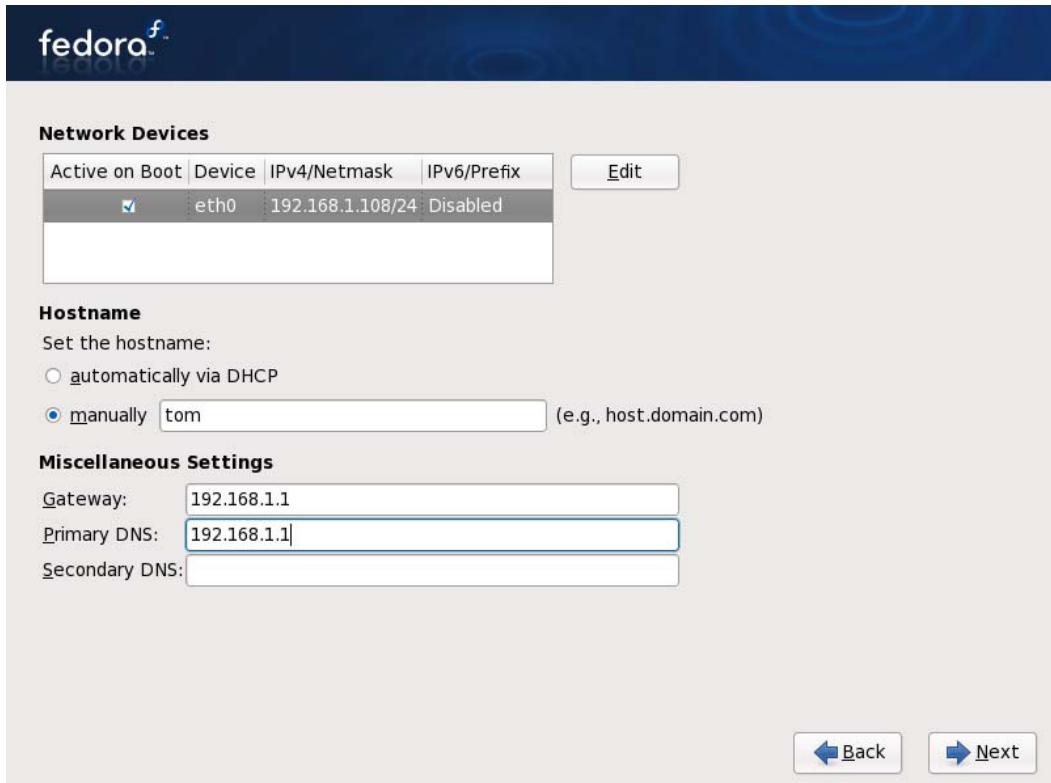
Step 6: configure the network.



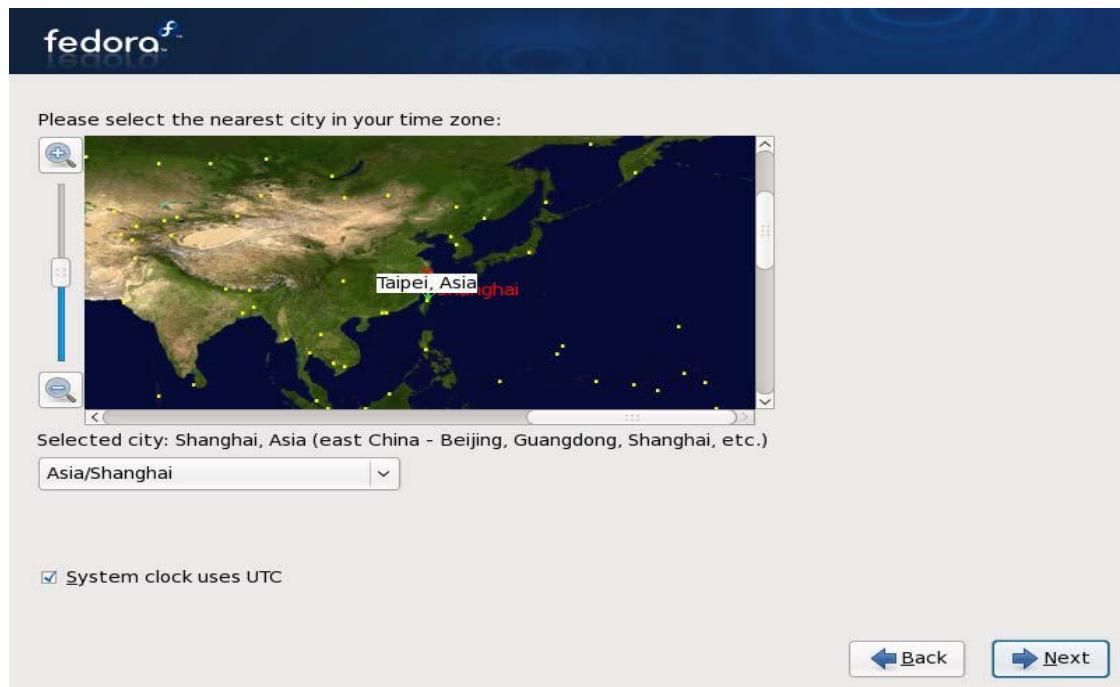
In our example, we didn't set it as "DHCP", we used a static IP instead, and typed the IP and subnet mask as follows.



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

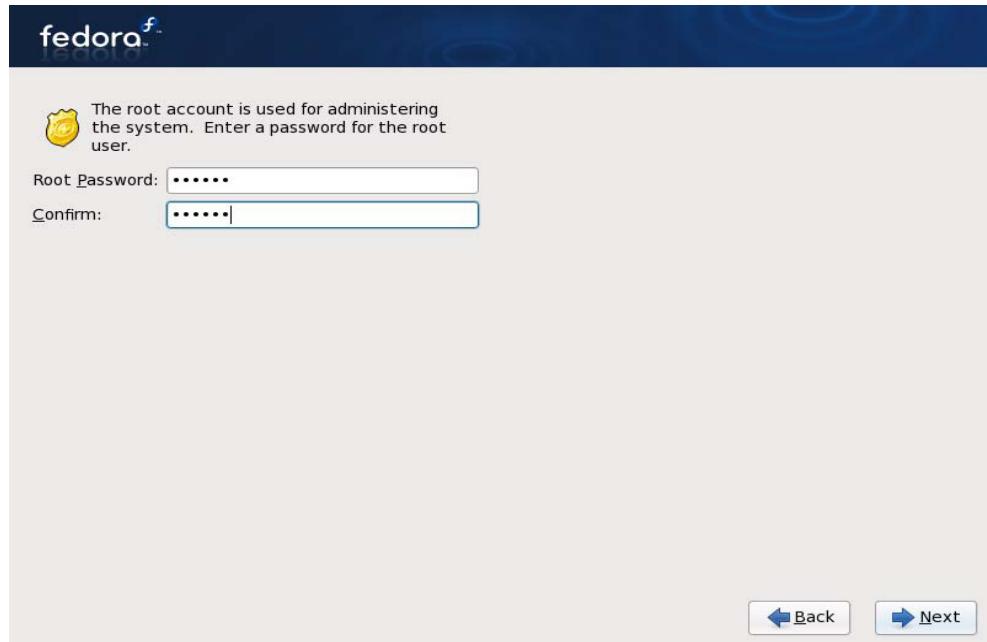


Step 7: set the time zone. We chose “Asia/Shanghai”.

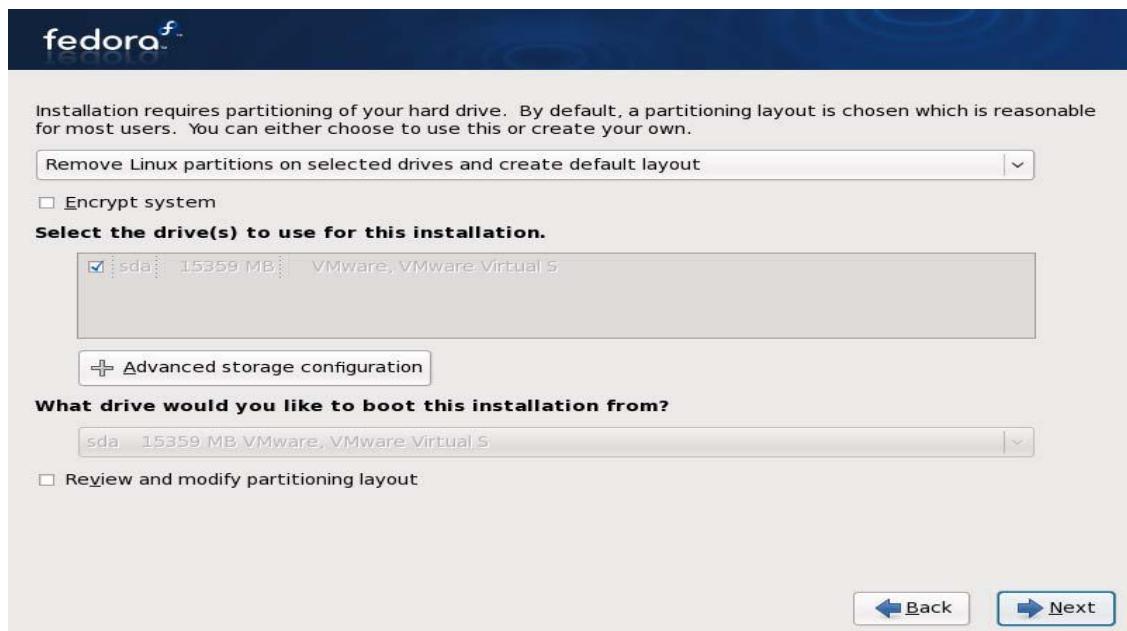


Step 8: set up the administrator's password, i.e. the root's password. "root" is the super user.

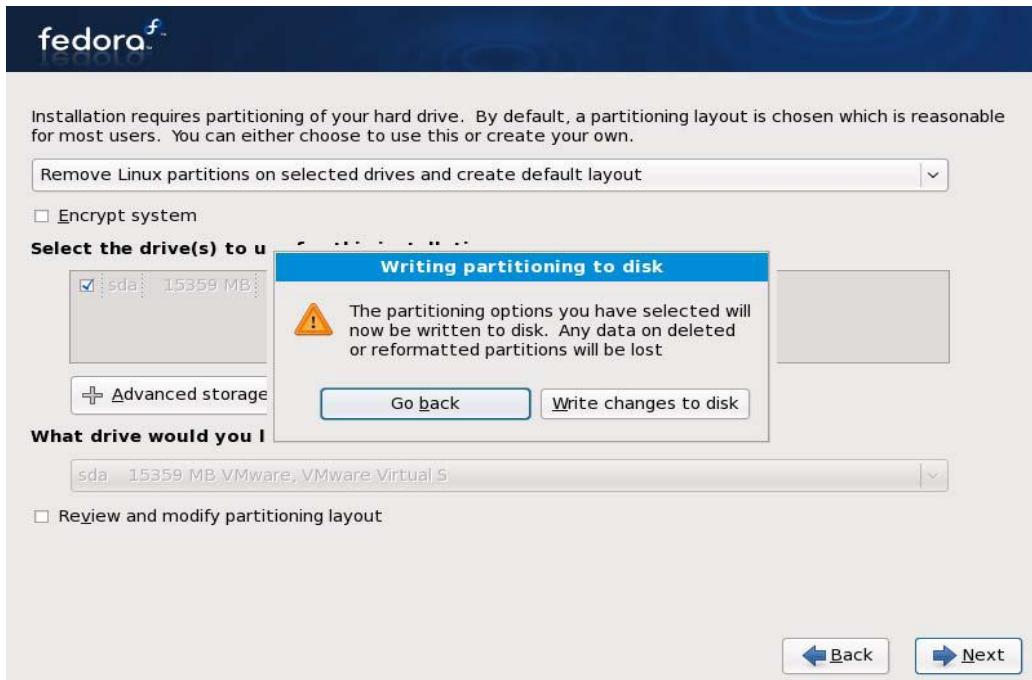
It should be at least 6 characters



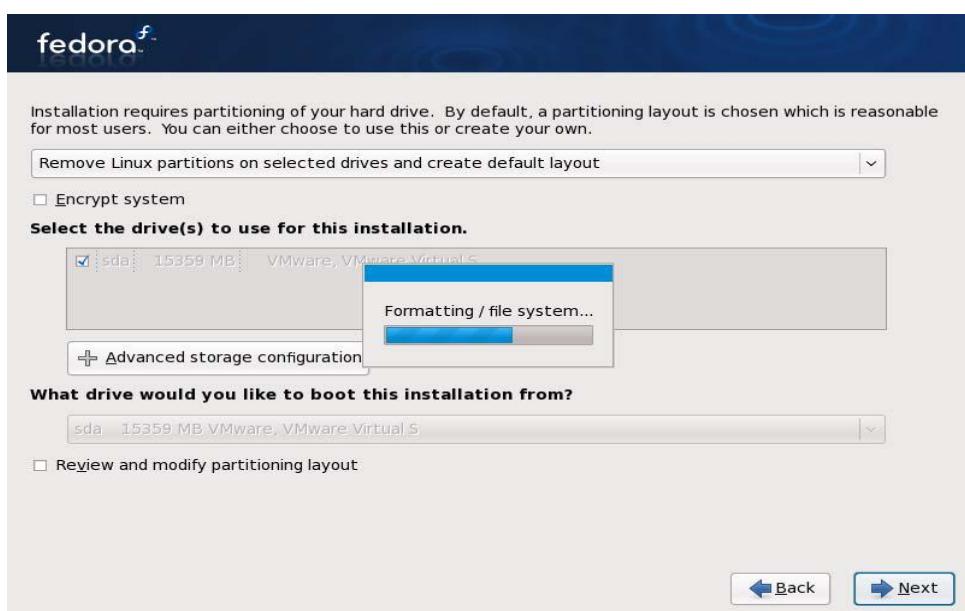
Step 9: disk partition. We followed the default option. Before do this, please back up disk data.



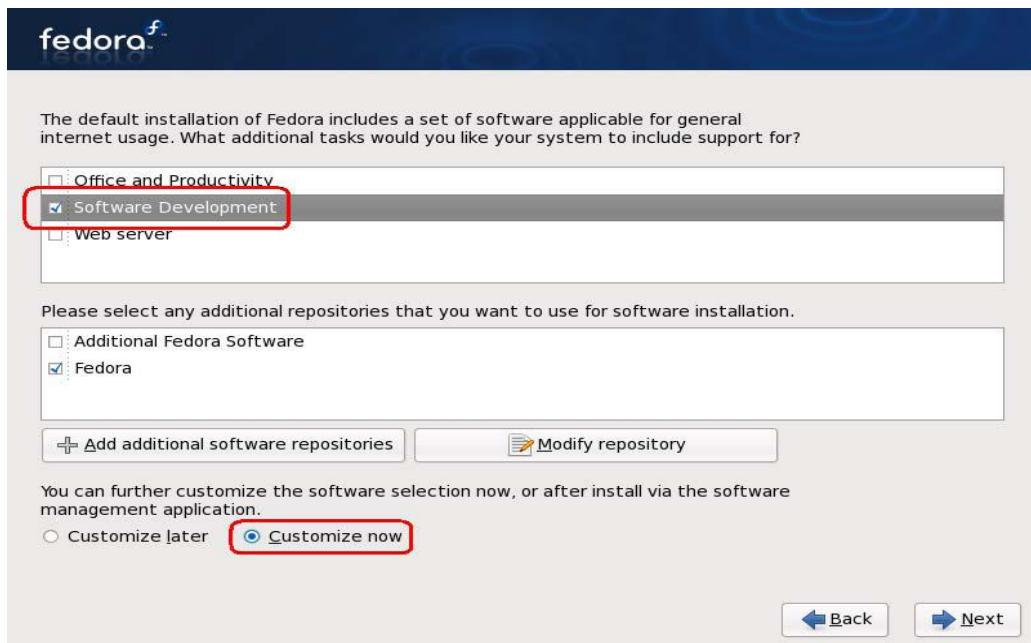
Click on “Next”, it will warn the user that all the data will be deleted. Usually we would do this installation in VMWARE, so we chose “Write changes to disk” and disk format would begin.



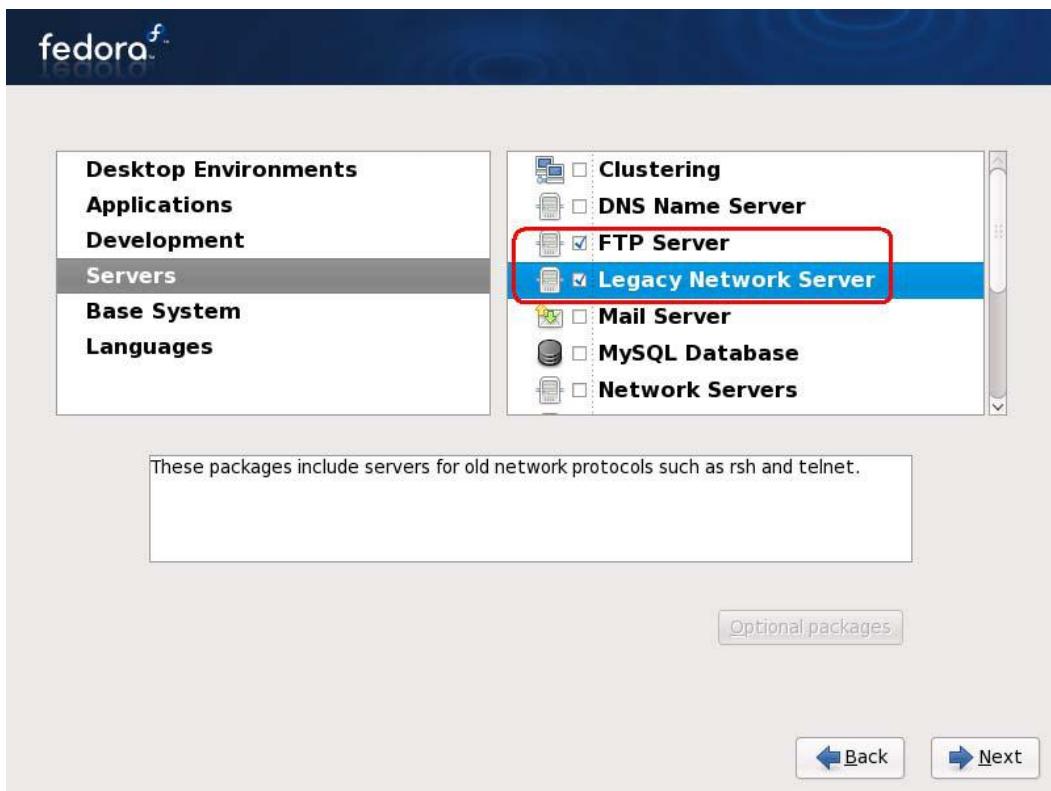
Here is the format process:



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



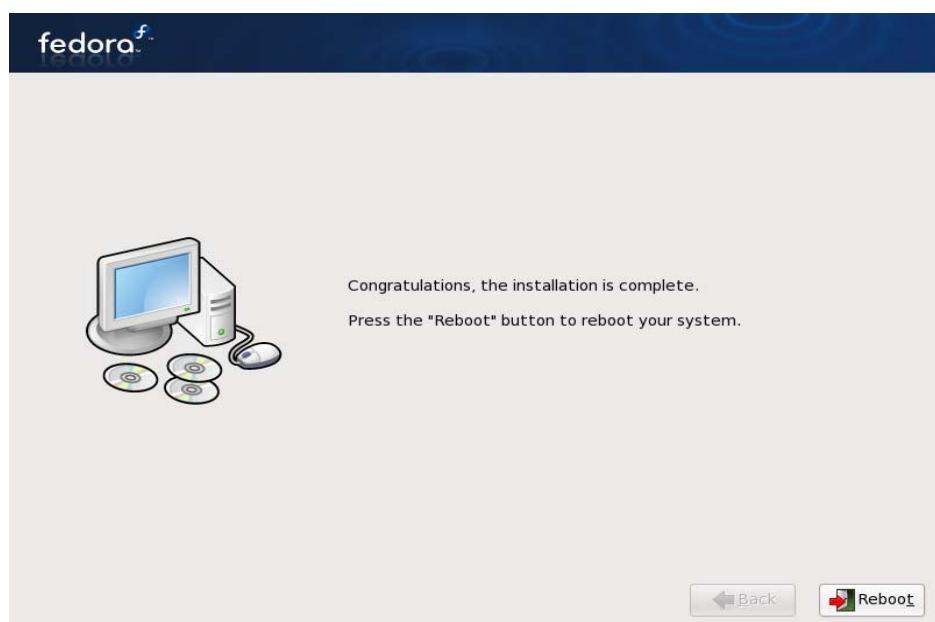
Step 12: configure the “server” item as follows:



Step 13: begin installation



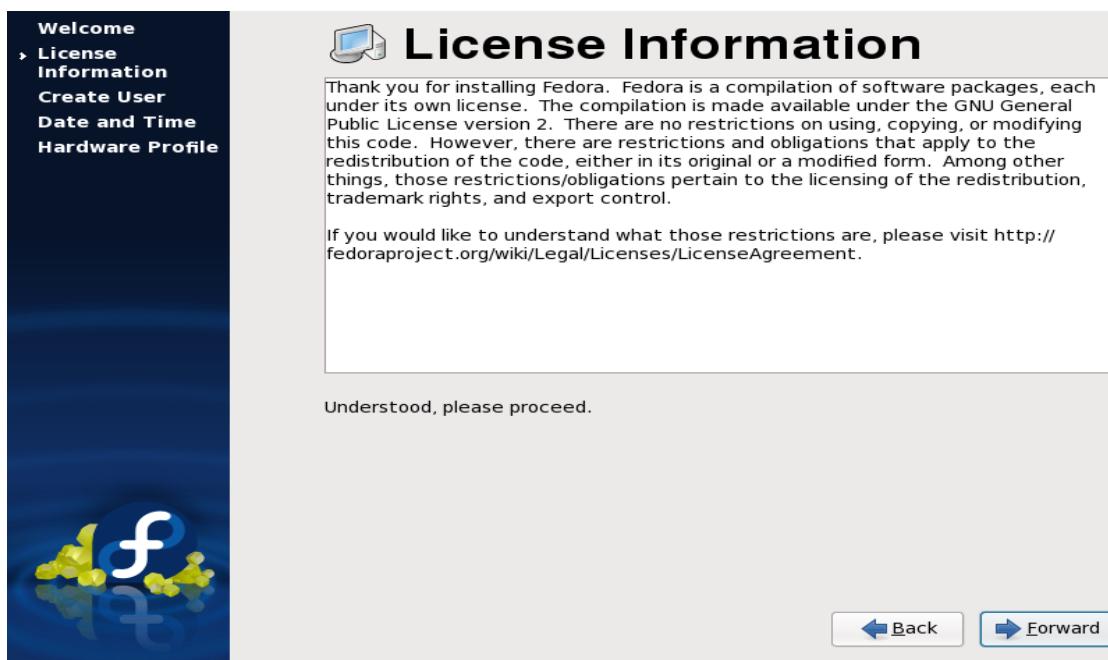
Step14: installation complete.



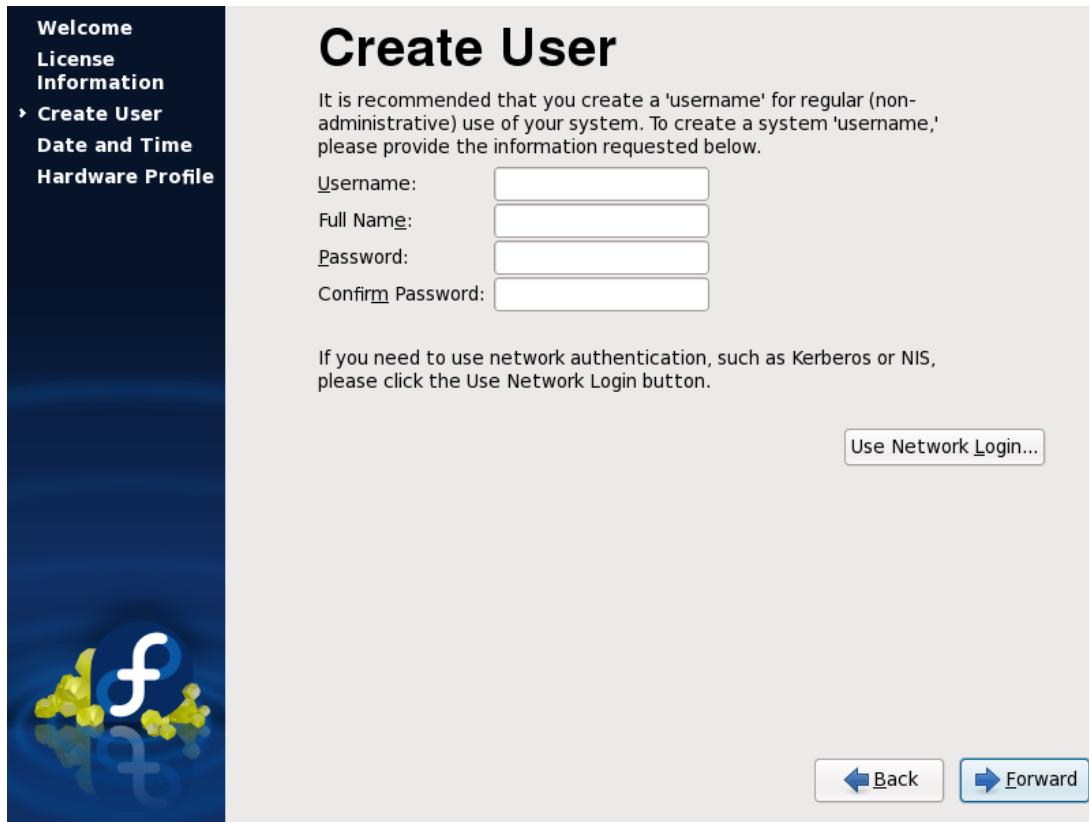
Step15: after installation completed, click on the reboot button on the page shown in step 14



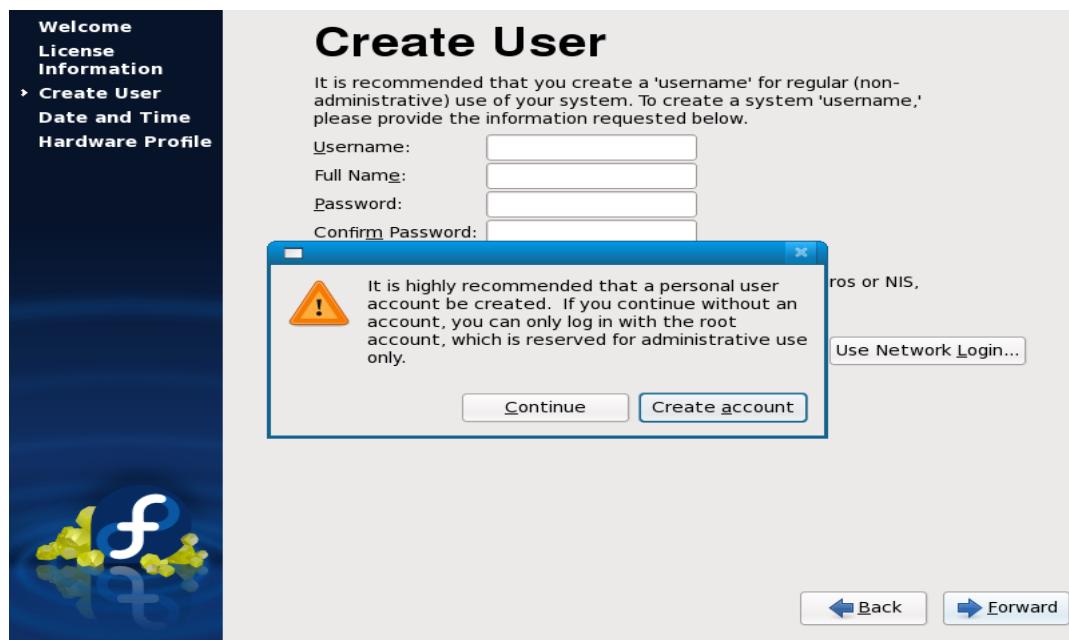
Step16: skip this license page and go “forward”



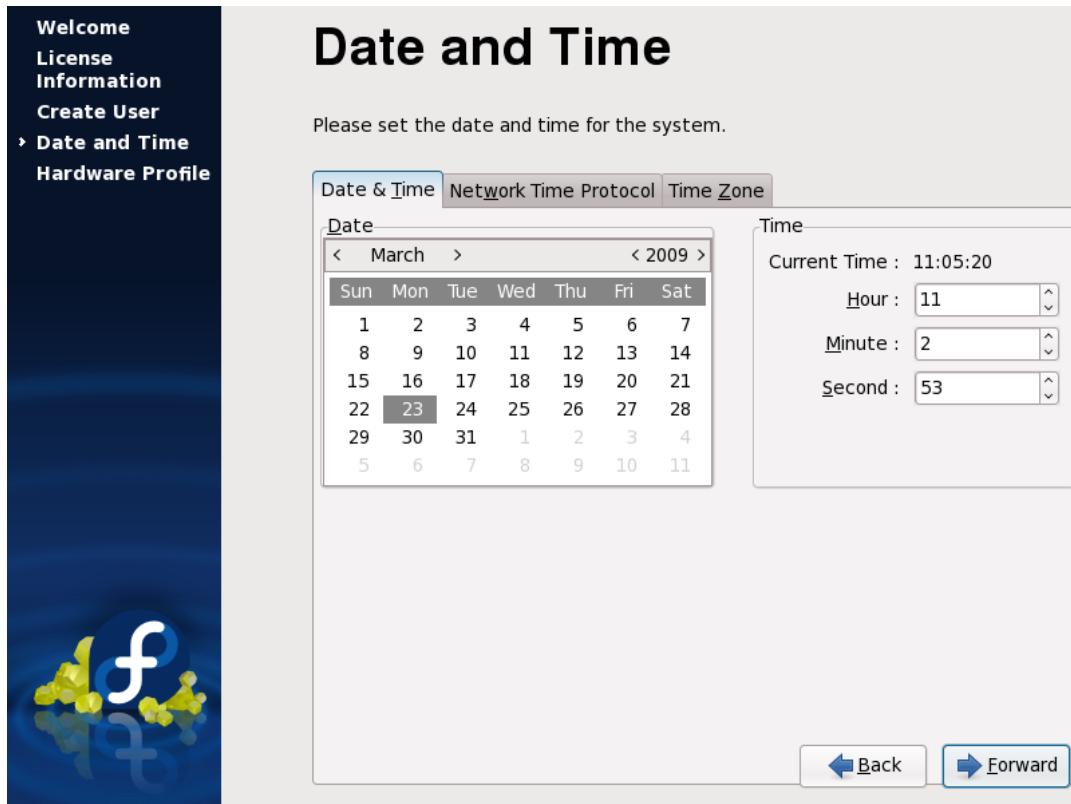
Step17: create new users. We ignored user creation and went to the next step.



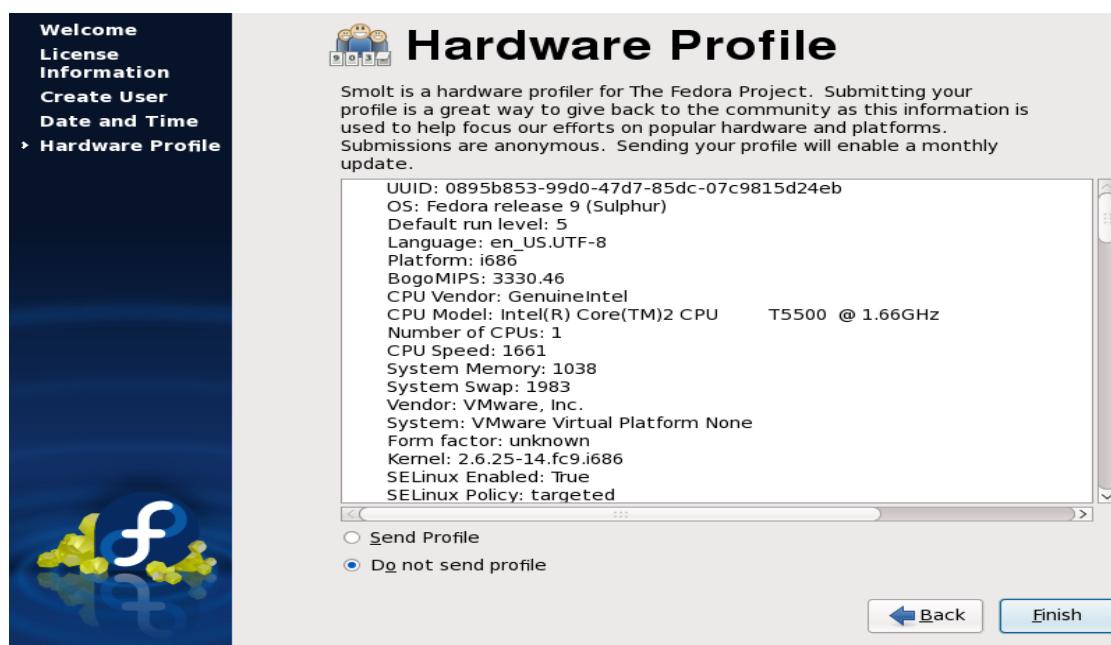
Press “continue” to go on.



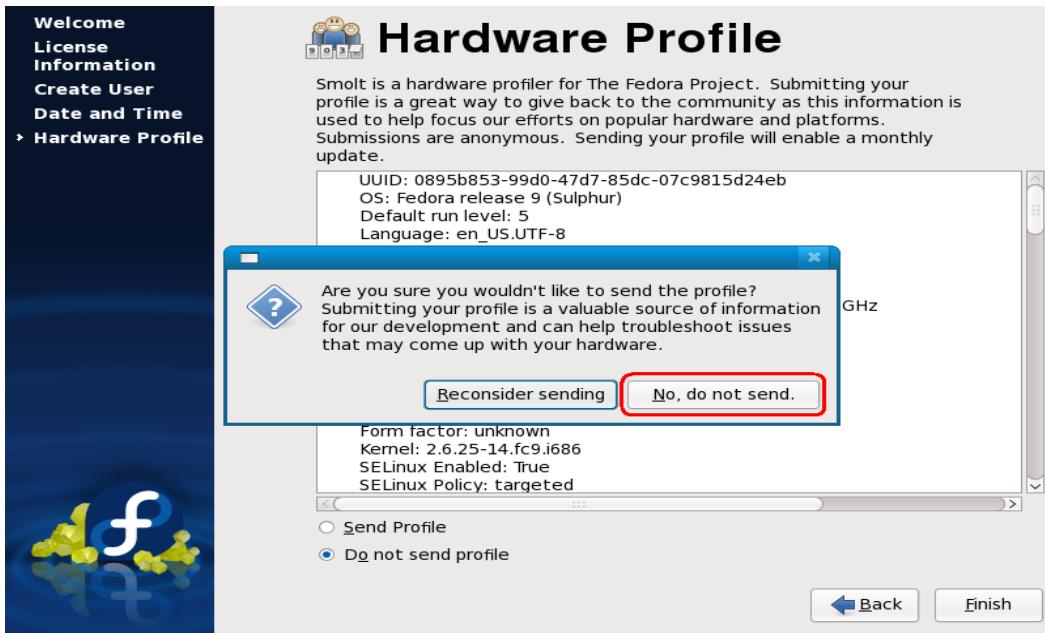
Step18: setup date and time. We ignored this and went to the next step.



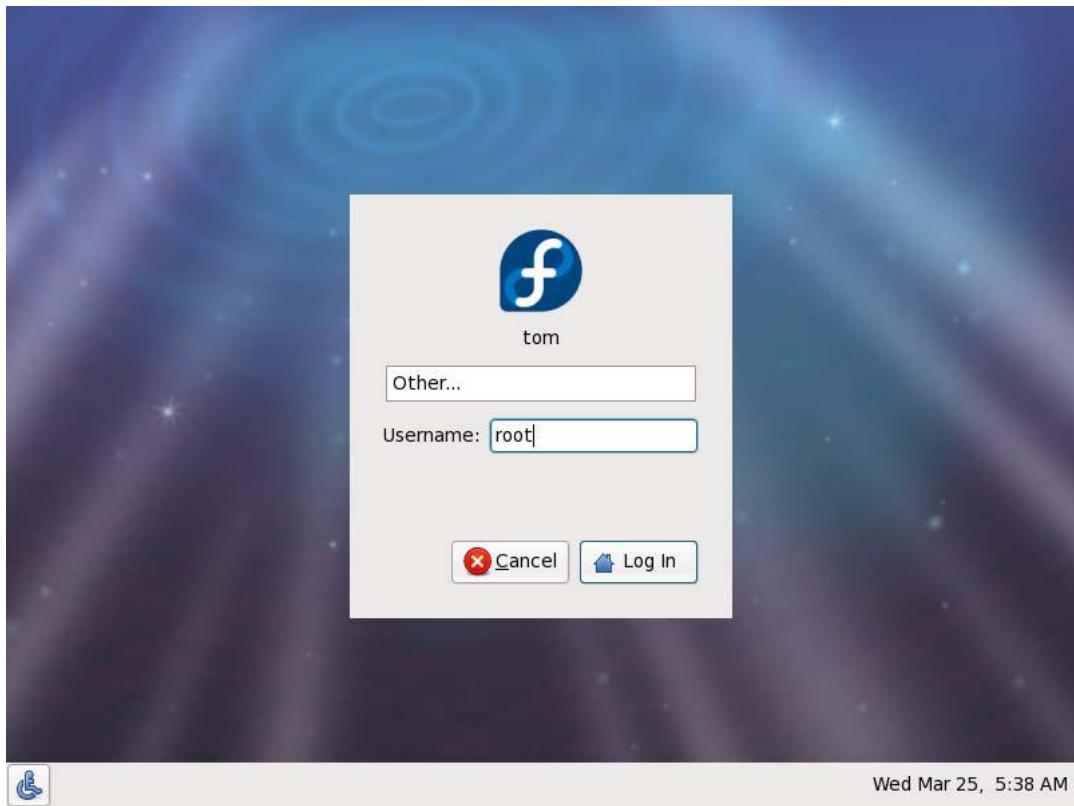
Step19: confirm hardware information. We just clicked on “Finish”.



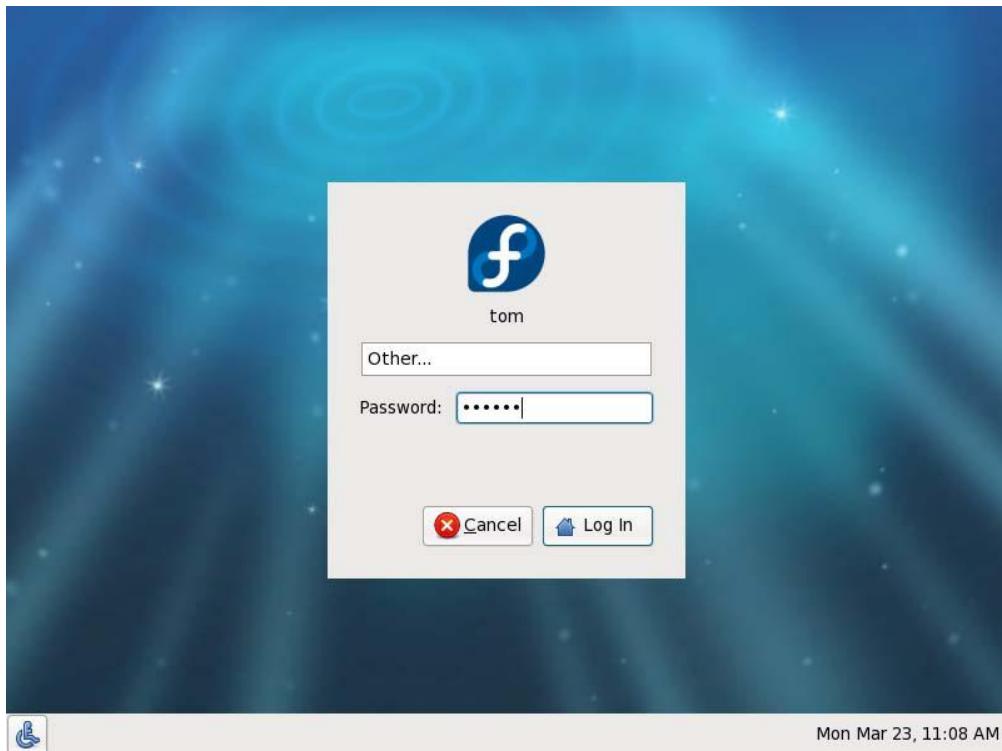
On the popup window shown below, just click on the red marked button.



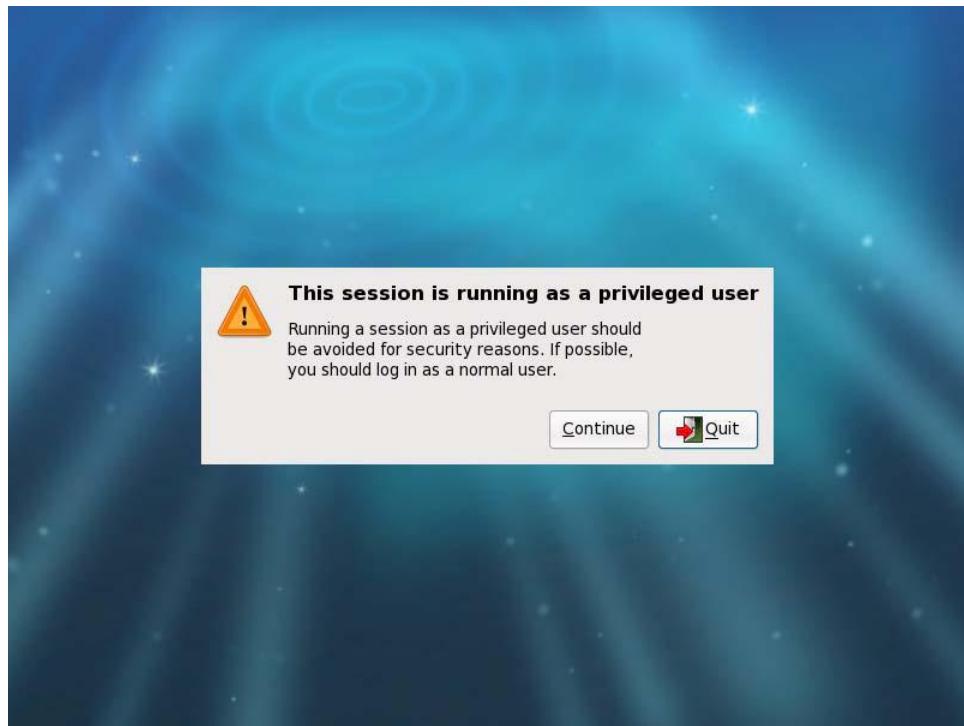
Step 20: on the login page, login as “root”



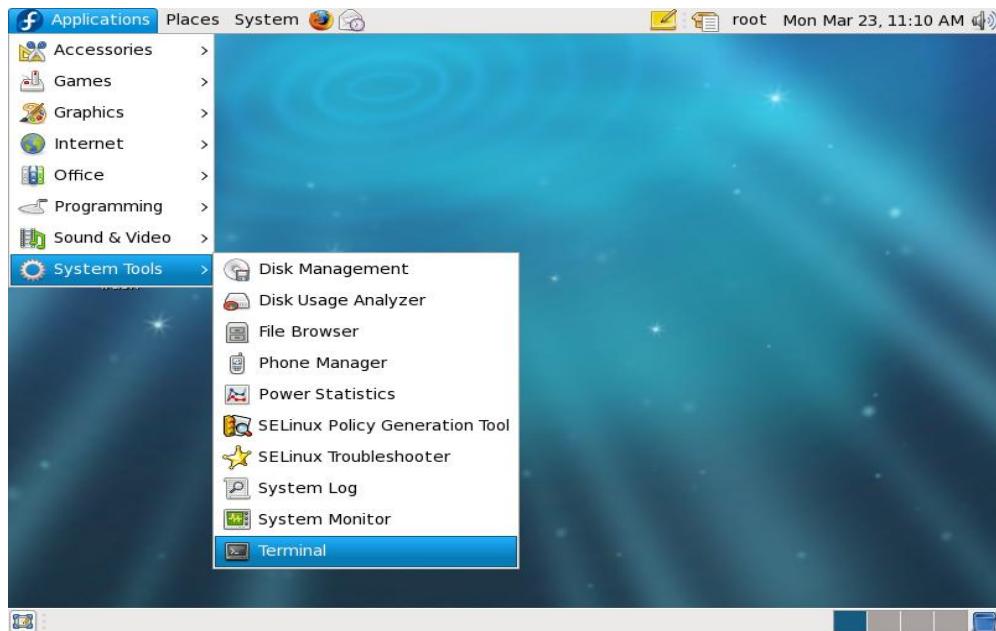
Input the password we just created for “root”



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



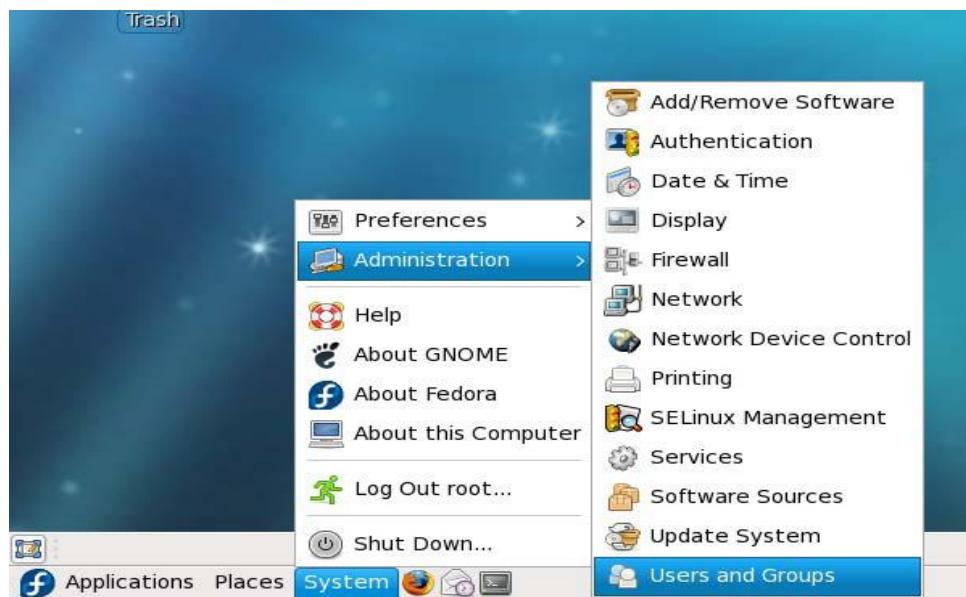
Below is the interface the user will see after a successful login.



4.3.2 Add User Account

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

Step 1: go to “Users and Groups”

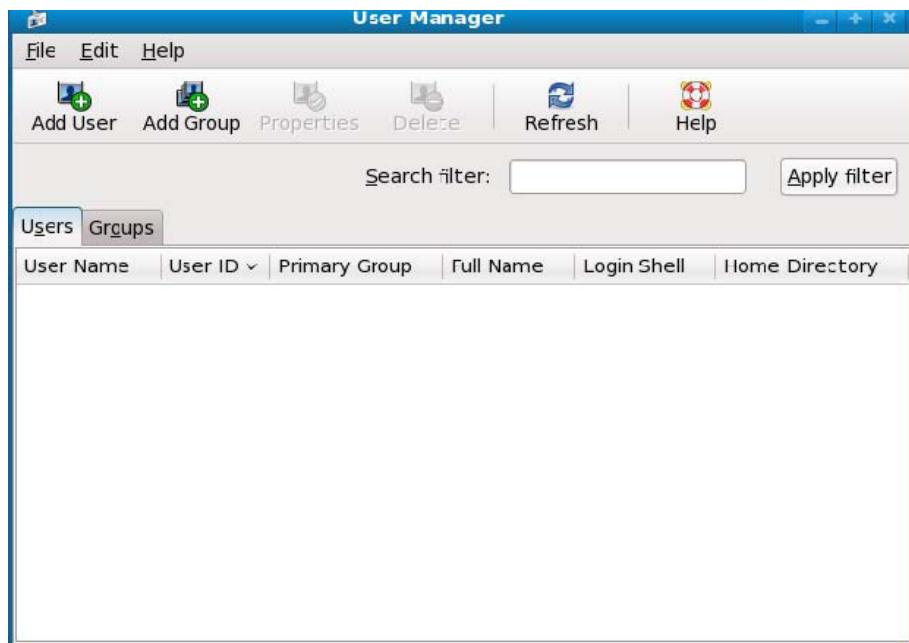


Address: Room 1701, Block A2, Longyuan Plaza, 549# Longkouxi Road, Guangzhou, China, 510640

Website: <http://www.arm9.net> Email: capbily@163.com

Tel: +86-20-85201025 Fax: +86-20-85261505

Step 2: open the “Users Manager” window

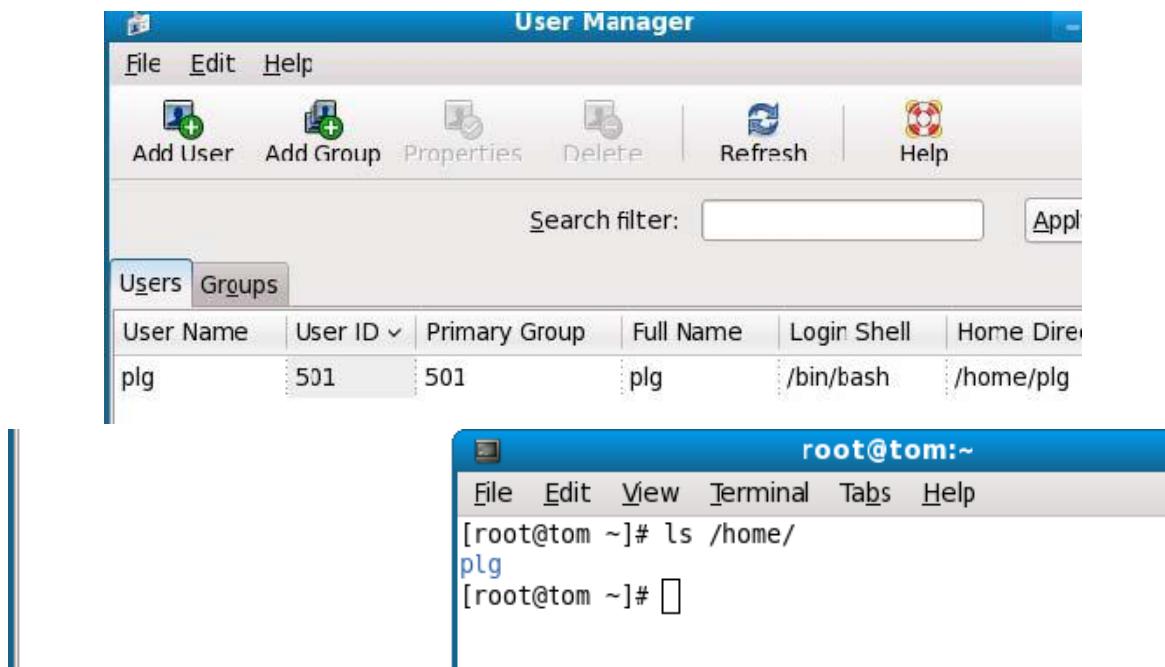


Step 3: click on the “Add User” button, type the user name and password



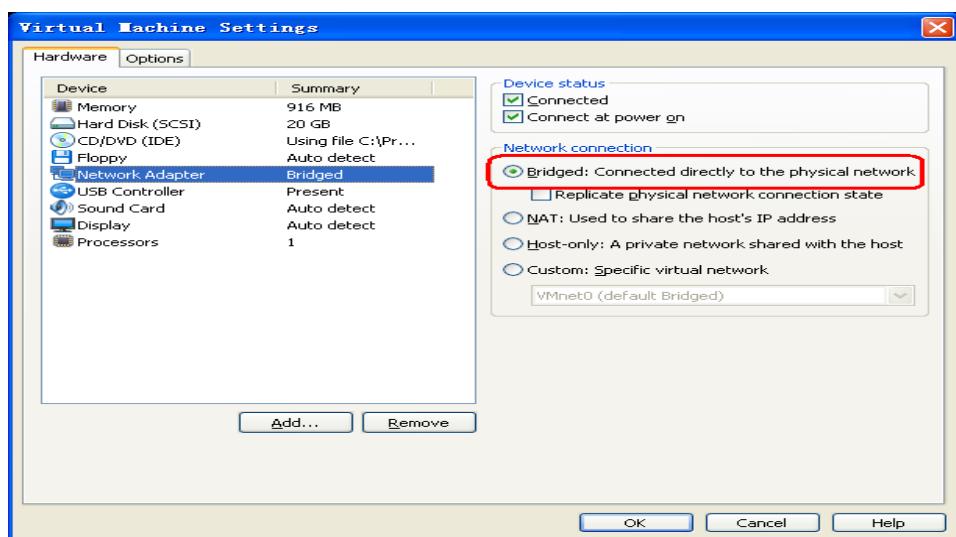
Click on “OK”, you will see that a new “plg” user has been created, and a “plg” directory has

been created in the “/home” directory too.



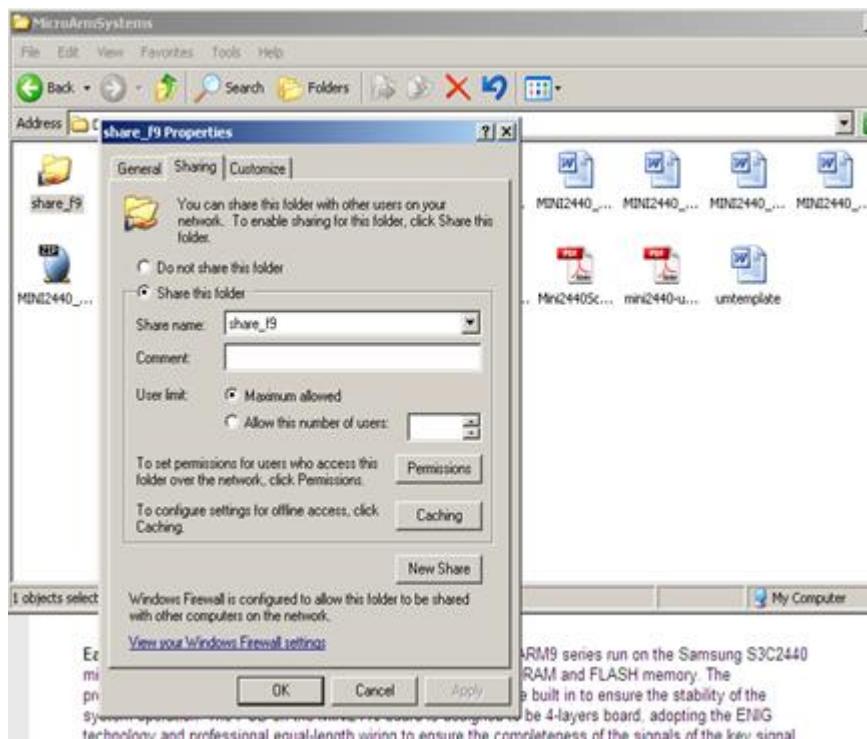
4.3.3 Access Windows Files

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.

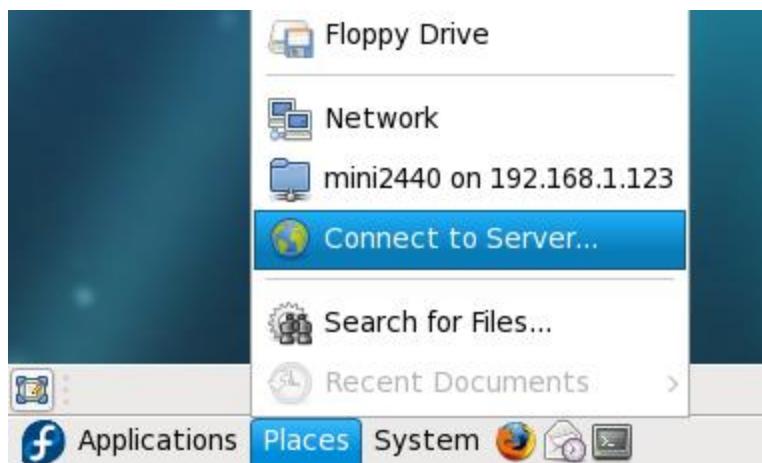


To access shared files in Windows, please follow the steps below:

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



Step 2: set Fedora9



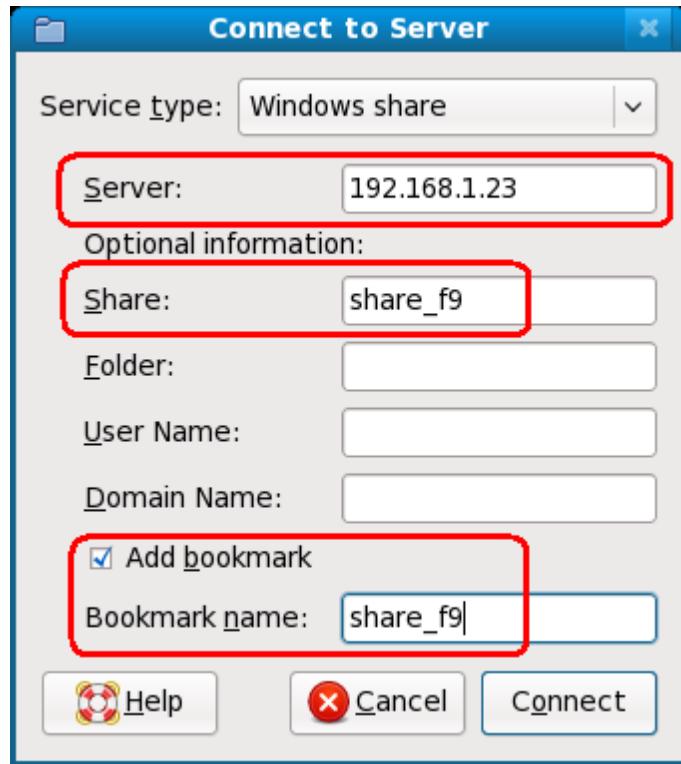
Open the window below:



Select "Windows share" in the "service type" field



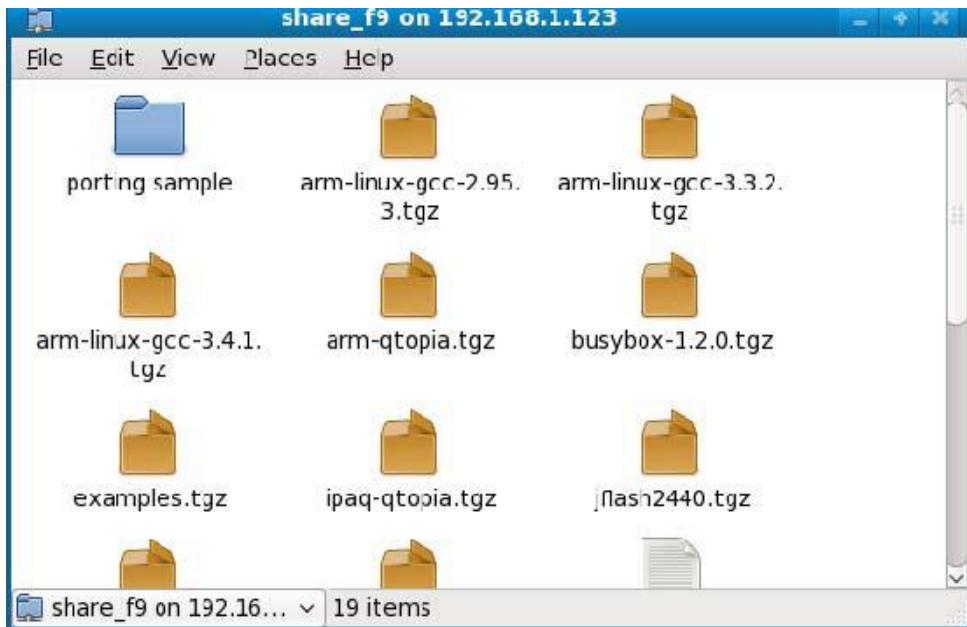
Input the shared file's name and its windows machine IP



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



Go ahead and “connect” again you will see the shared files.



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

```

root@tom ~]# ls /root/.gvfs/
mini2440 on 192.168.1.123/ share_f9 on 192.168.1.123/
[root@tom ~]# ls /root/.gvfs/share_f9 on 192.168.1.123/
arm-linux-gcc-2.95.3.tgz      porting sample
arm-linux-gcc-3.3.2.tgz      readme.txt
arm-linux-gcc-3.4.1.tgz      root_default.tgz
arm-qtopia.tgz               root_mizi.tgz
busybox-1.2.0.tgz           root_nfs.tgz
examples.tgz                 root_qtopia_mouse.tgz
ipaq-qtopia.tgz              root_qtopia_tp.tgz
jflash2440.tgz               vivi.tgz
kernel-2.6.13-mini2440-20081127.tgz x86-qtopia.tgz
mkyaffsimage.tgz
[root@tom ~]# 

```

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



4.3.4 Set up 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 5.4.2 to install the root_qtopia 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/root_qtopia_qt4
```

*(rw, sync, no_root_squash)

“/opt/FriendlyARM/mini6410/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 its 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/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/root_qtopia_qt4” directory.

Stop the service by commanding:

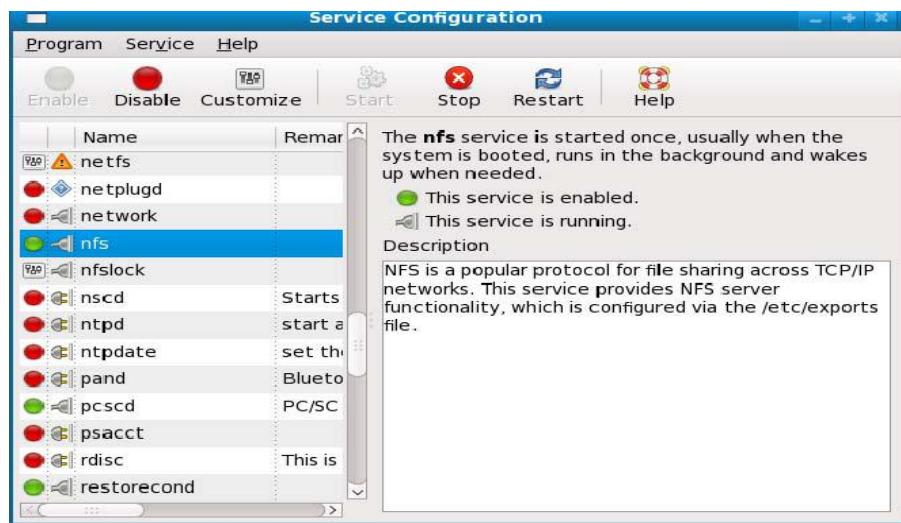
```
#/etc/init.d/nfs stop
```

(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/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.

```
[1] 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 (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 I0 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 /]# _
```

```
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 I0 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 /]# _
```

4.3.5 Set up Cross Compile Environment

To compile kernels, Qtopia/Qt4, bootloader and other programs in Linux you need a cross compile environment. We used arm-linux-gcc-4.5.1 and its by default 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.

```
root@tom:/opt/FriendlyARM/toolschain/4.5.1
File Edit View Terminal Tabs Help
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

export PATH=$PATH:/opt/FriendlyARM/toolschain/4.5.1/bin
~
```

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

```
root@tom:/opt/FriendlyARM/toolschain/4.5.1
File Edit View Terminal Tabs Help
[root@tom 4.5.1]# arm-linux-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolschain/4.5.1/libexec/gcc/arm-none-linux-gnueabi/4.5.1/lto-wrapping
Target: arm-none-linux-gnueabi
Configured with: /work/toolchain/build/src/gcc-4.5.1/configure --build=i686-build_pc-linux-gnu --host=i686-build_pc-linux-gnu --target=arm-none-linux-gnueabi --prefix=/opt/FriendlyARM/toolschain/4.5.1 --with-sysroot=/opt/FriendlyARM/toolschain/4.5.1/arm-none-linux-gnueabi/sys-root --enable-languages=c,c++ --disable-multilib --with-cpu=arm1176jzf-s --with-tune=arm1176jzf-s --with-fpu=vfp --with-float=softfp --with-pkgversion=ctng-1.8.1-FA --with-bugurl=http://www.arm9.net/ --disable-sjlj-exceptions --enable_cxa_atexit --disable-libmudflap --with-host-libstdcxx='static-libgcc -Wl,-Bstatic,-Lstdc++,-Bdynamic -lm' --with-gmp=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-mpfr=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-ppl=/work/toolchain/build/arm-non-e-linux-gnueabi/build/static --with-cloog=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-mpc=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-libelf=/work/toolchain/build/arm-none-linux-gnueabi/build/static --enable-threads=posix --with-local-prefix=/opt/FriendlyARM/toolschain/4.5.1/arm-none-linux-gnueabi/sys-root --disable-nls --enable-symvers=gnu --enable-c99 --enable-long-long
Thread model: posix
gcc version 4.5.1 (ctng-1.8.1-FA)
[root@tom 4.5.1]#
```

4.4 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**

4.4.1 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 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-20100420.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 be 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.

4.4.2 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 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)

4.4.3 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.

For the Mini6410-1405 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** which is especially for UBIFS. 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 “mkyaffs2image”, “mkyaffs2image-128M”, “mkubimage” and “mkext3image” 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

4.4.4 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



4.5 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

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

4.5.1 Compute 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.

4.5.2 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.

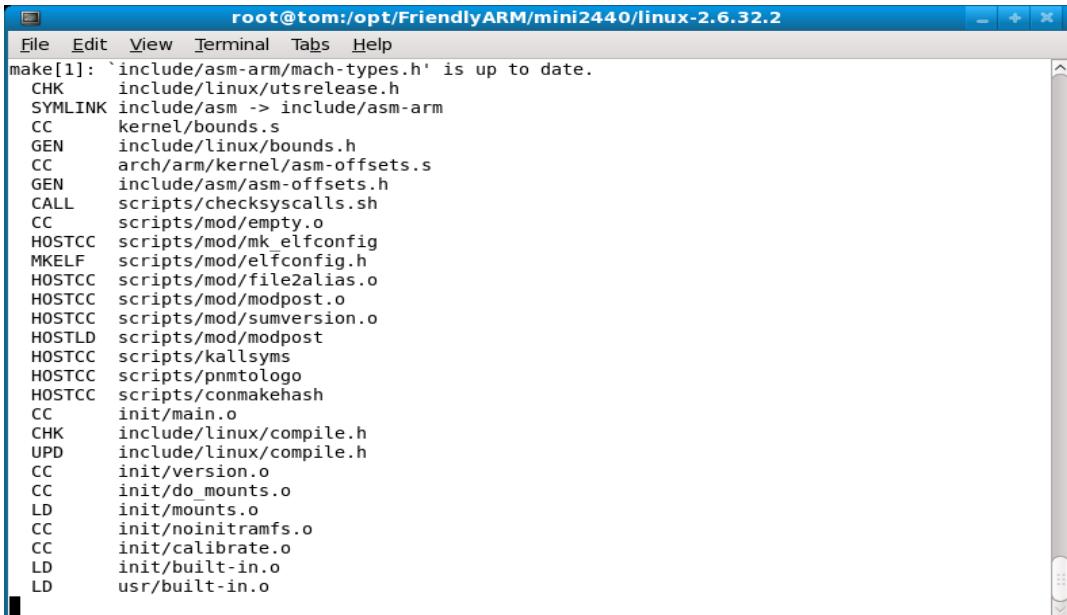
4.6 Compile Kernel

4.6.1 Compile Kernel

Type the following command to compile:

```
#cp config_mini6410 .config : there is a "." prior to "config"
```

#make zImage ; begins to compile



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.32.2
File Edit View Terminal Tabs Help
make[1]: `include/linux/mach-types.h' is up to date.
CHK include/linux/utsrelease.h
SYMLINK include/arm -> include/arm-arm
CC kernel/bounds.s
GEN include/linux/bounds.h
CC arch/arm/kernel/asm-offsets.s
GEN include/arm/asm-offsets.h
CALL scripts/checksyscalls.sh
CC scripts/mod/empty.o
HOSTCC scripts/mod/mk_elfconfig
MKELF scripts/mod/elfconfig.h
HOSTCC scripts/mod/file2alias.o
HOSTCC scripts/mod/modpost.o
HOSTCC scripts/mod/sumversion.o
HOSTLD scripts/mod/modpost
HOSTCC scripts/kallsyms
HOSTCC scripts/pnmtologo
HOSTCC scripts/conmakehash
CC init/main.o
CHK include/linux/compile.h
UPD include/linux/compile.h
CC init/version.o
CC init/do_mounts.o
LD init/mounts.o
CC init/noinitramfs.o
CC init/calibrate.o
LD init/built-in.o
LD usr/built-in.o
```

After the compilation is done, an image file **zImage** will be generate under
“arch/arm/boot”.

4.6.2 Drivers

Here we listed the drivers under the Linux system

| | Device | Driver | Device Name | Comment |
|---|--------------------|---|-------------------------|--------------------------|
| 1 | yaffs2 file system | Linux-2.6.38/fs/yaffs2 | | |
| 2 | ubifs file system | Linux-2.6.38/fs/ubifs | | |
| 3 | LCD framebuffer | Linux-2.6.38/drivers/video/Samsung/s3c_mini6410.c | /dev/fb0 | |
| 4 | Serial port | Linux-2.6.38/drivers/tty/serial/Samsung.c | /dev/ttySAC0,1,2,3 | 6410's four serial ports |
| 5 | Ethernet | Linux-2.6.38/drivers/net/dm9000.c | | |
| 6 | Audio(ALS) | Linux-2.6.38/sound/soc/codecs/wm9713.c | /dev/dsp: audio playing | |



| | | | | |
|----|------------------------|--|--|---|
| | A) | | and recording /dev/mixer: volumn | |
| 7 | ARM Touchscreen | Linux-2.6.38/drivers/input/touchscreen/mini6410_ts.c | /dev/touchscreen | |
| 8 | One wire precise touch | Linux-2.6.38/drivers/input/touchscreen/mini6410_1wire_host.c | /dev/touchscreen-1wire | |
| 9 | SD card | Linux-2.6.38/drivers/mmc/host/sdhci-s3c.c | /dev/mmcblk0 | High speed, large volumn SD card up to 32G |
| 10 | RTC real time clock | Linux-2.6.38/drivers/rtc/rtc-s3c.c | /dev/rtc | |
| 11 | Watchdog | Linux-2.6.38/drivers/watchdog/s3c2410_wdt.c | /dev/watchdog | |
| 12 | LED | Linux-2.6.38/drivers/char/mini6410_leds.c | /dev/leds | |
| 13 | User button | Linux-2.6.38/drivers/char/mini6410_buttons.c | /dev/buttons | |
| 14 | PWM | Linux-2.6.38/drivers/char/mini6410_pwm.c | /dev/pwm | |
| 15 | ADC | Linux-2.6.38/drivers/char/mini6410_adc.c | /dev/adc | |
| 16 | LCD backlight | Linux-2.6.38/drivers/video/mini6410_backlight.c | /dev/backlight-1wire | |
| 17 | I2C-EEPROM | Linux-2.6.38/drivers/i2c/busses/i2c-s3c2410.c | /dev/i2c/0 | |
| 18 | USB camera | Linux-2.6.38/drivers/media/video/gspca | /dev/video0 | |
| 19 | USB WiFi | Linux-2.6.38/drivers/net/wireless | | |
| 20 | SD WiFi | Linux-2.6.38/drivers/net/wireless/libertas | | |
| 21 | USB to Serial | Linux-2.6.38/drivers/usb/serial | /dev/ttyUSB0 | |
| 22 | USB device | Linux-2.6.38/drivers/usb/hid | USB mouse:/dev/input/mice USB keyboard: /dev/input | |
| 23 | SPI | Linux-2.6.38/drivers/spi/spi_s3c64xx.c | | |
| 24 | Multi-media | Linux-2.6.38/drivers/media/video/Samsung | | This is not open |

| | | | | source |
|----|------------------|--|--|--------|
| 25 | Nand Flash | Linux-2.6.38/drivers/mtd/nand | | |
| 26 | Flash ECC | Linux-2.6.38/drivers/mtd/nand/s3c_nand.c | | |
| 27 | USB bluetooth | Linux-2.6.38/drivers/bluetooth/ | | |
| 28 | 3G | Linux-2.6.38/drivers/usb/serial | | |

4.7 Compile Busybox

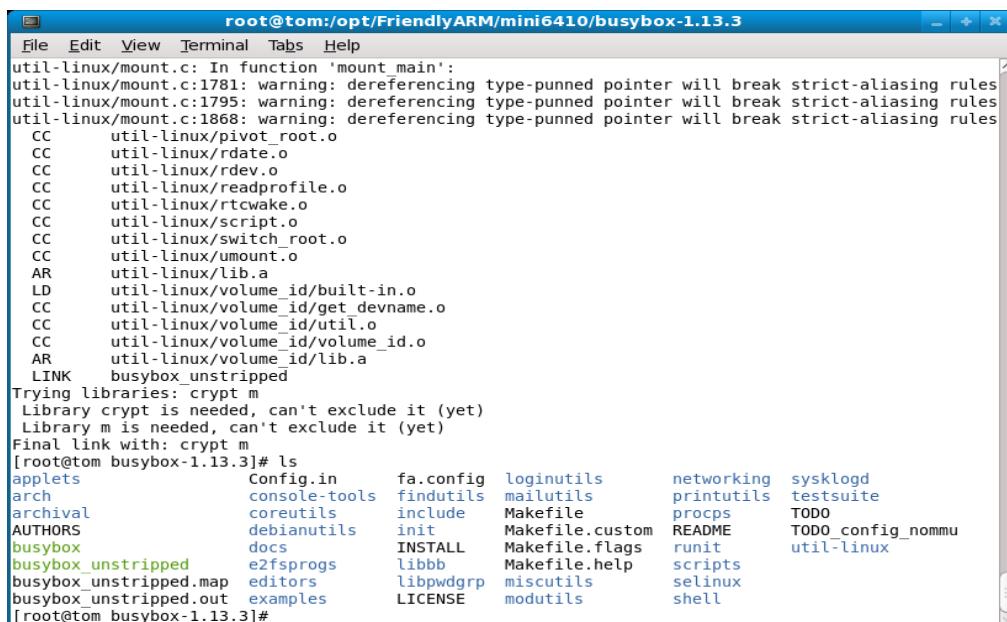
In general a direct download of busybox from its website may be compiled through and its configurations need to be modified. We made one for users: fa.config. Both our 2440 and 6410 systems use this file. Please follow the steps below to install.

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



```
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]#
```

4.8 Make File System Image

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

4.8.1 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

4.8.2 Make UBIFS Image

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

```
#mkyaffs2image-128M rootfs_qtopia_qt4 rootfs_qtopia_qt4.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

4.8.3 Make EXT3 Image

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

```
#mkimage -t ext3 -F -C none rootfs_qtopia_qt4 rootfs_qtopia_qt4.ext3
```

This will compress the whole “rootfs_qtopia_qt4” into a EXT3 rootfs_qtopia_qt4.img file.

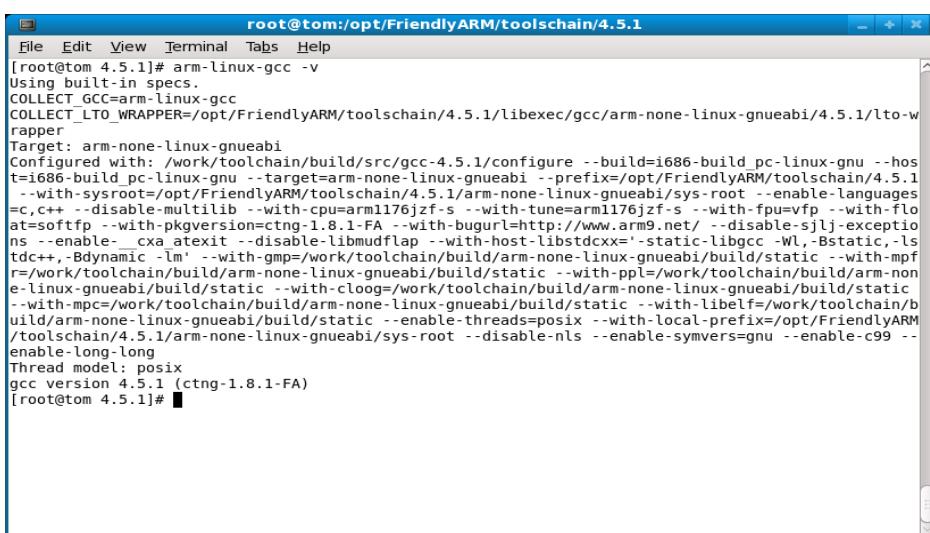
It is the same as the one in “/images/Linux/” in the shipped CD. Download it to your board’s NAND Flash.

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

4.9 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. To check your compiler, please type “arm-linux-gcc -v”



```
root@tom:~/opt/FriendlyARM/toolschain/4.5.1#
File Edit View Terminal Tabs Help
[root@tom 4.5.1]# arm-linux-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolschain/4.5.1/libexec/gcc/arm-none-linux-gnueabi/4.5.1/lto-wrapper
Target: arm-none-linux-gnueabi
Configured with: /work/toolchain/build/src/gcc-4.5.1/configure --build=i686-build_pc-linux-gnu --host=i686-build_pc-linux-gnu --target=arm-none-linux-gnueabi --prefix=/opt/FriendlyARM/toolschain/4.5.1 --with-sysroot=/opt/FriendlyARM/toolschain/4.5.1/arm-none-linux-gnueabi/sys-root --enable-languages=c,c++ --disable-multilib --with-cpu=arm1176zf-s --with-tune=arm1176zf-s --with-fpu=vfp --with-float=softfp --with-pkgversion=ctng-1.8.1-FA --with-bugurl=http://www.arm9.net/ --disable-sjlj-exceptions --enable_cxa_atexit --disable-libmudflap --with-host-libstdcxx=static-libgcc -Wl,-Bstatic,-ls -tdc++, -Bdynamic -lm' --with-gmp=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-mpfr=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-ppl=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-cloog=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-mpc=/work/toolchain/build/arm-none-linux-gnueabi/build/static --with-libelf=/work/toolchain/build/arm-none-linux-gnueabi/build/static --enable-long-long
Thread model: posix
gcc version 4.5.1 (ctng-1.8.1-FA)
[root@tom 4.5.1]#
```

The following programs are developed by Friendly ARM. We have found that some companies illegally copied our programs.

4.9.1 “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");

}
```

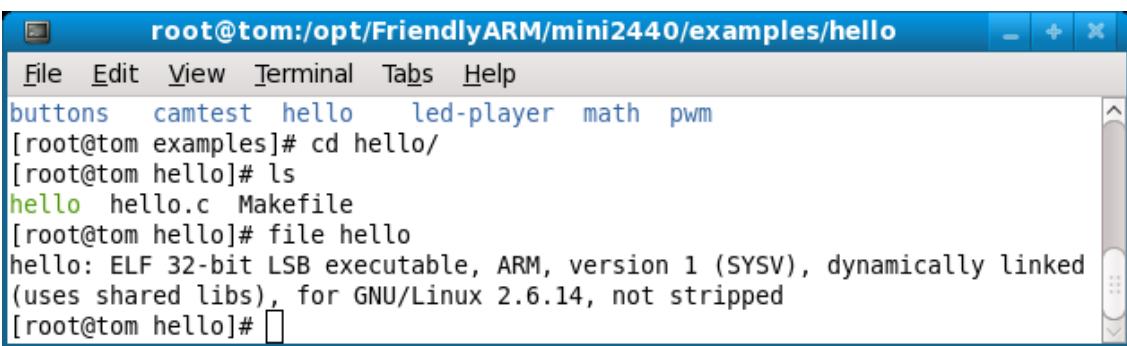
Step1: Compile Hello,World

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”:



```
root@tom:/opt/FriendlyARM/mini2440/examples/hello
File Edit View Terminal Tabs Help
buttons camtest hello led-player math pwm
[root@tom examples]# cd hello/
[root@tom hello]# ls
hello hello.c Makefile
[root@tom hello]# file hello
hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked
(uses shared libs), for GNU/Linux 2.6.14, not stripped
[root@tom hello]#
```

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

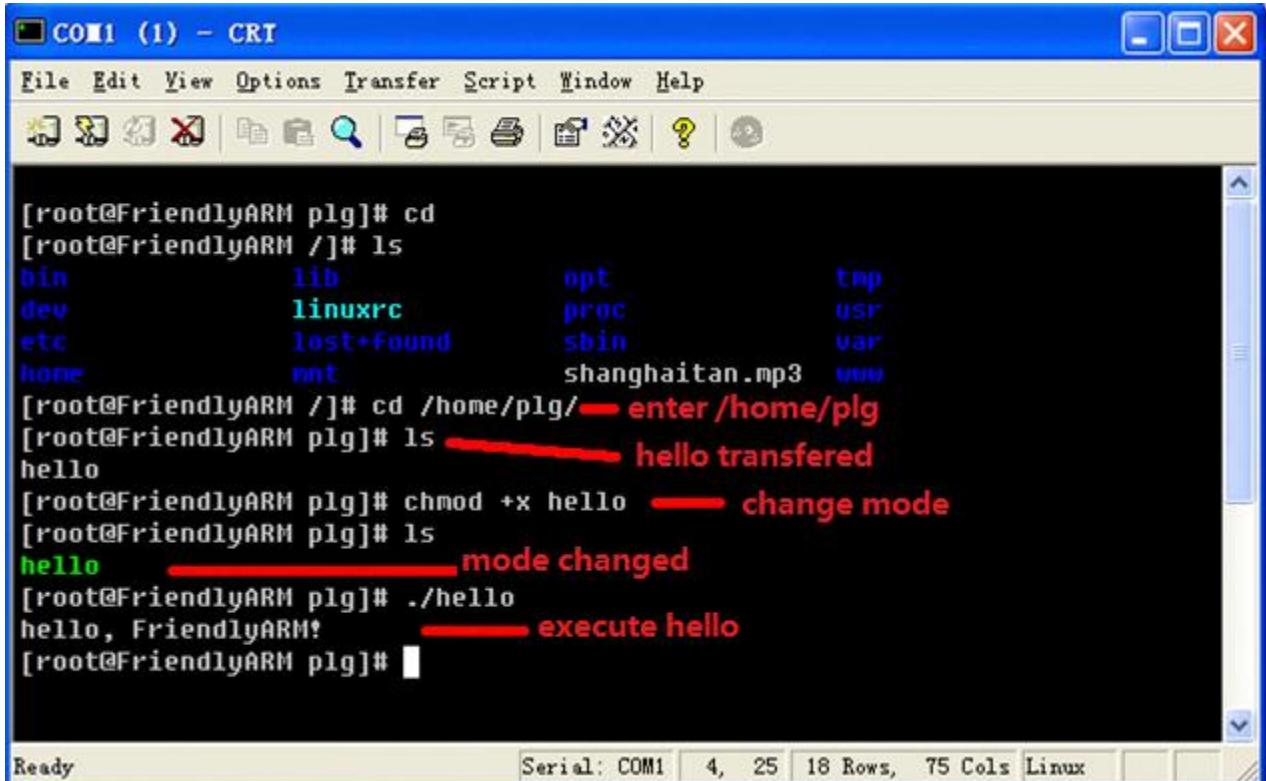
First, execute your commands in PC

```

root@tom:/opt/FriendlyARM/mini2440/examples/hello
File Edit View Terminal Tabs Help
[root@tom hello]# ls
hello hello.c Makefile
[root@tom hello]# ftp 192.168.1.230 1. Login
Connected to 192.168.1.230 (192.168.1.230).
220 FriendlyARM FTP server (Version 6.4/OpenBSD/Linux-ftp-0.17) ready.
Name (192.168.1.230:root): plg 2. Type name and password
331 Password required for plg.
Password:
230 User plg logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bin 3. Set file transfer format
200 Type set to I.
ftp> put hello 4. Upload hello
local: hello remote: hello
227 Entering Passive Mode (192,168,1,230,171,47)
150 Opening BINARY mode data connection for 'hello'.
226 Transfer complete.
5061 bytes sent in 0.000144 secs (35145.83 Kbytes/sec)
ftp> by 5. Logout
221 Goodbye.
[root@tom hello]#

```

Go to your board and execute the following commands:



```
[root@FriendlyARM plg]# cd
[root@FriendlyARM /]# ls
bin          lib          opt          tmp
dev          linuxrc      proc         usr
etc          lost+found   sbin         var
home         mnt         shanghai.mp3  www
[root@FriendlyARM /]# cd /home/plg/ — enter /home/plg
[root@FriendlyARM plg]# ls — hello transferred
hello
[root@FriendlyARM plg]# chmod +x hello — change mode
[root@FriendlyARM plg]# ls
hello — mode changed
[root@FriendlyARM plg]# ./hello — execute hello
hello, FriendlyARM!
[root@FriendlyARM plg]#
```

Ready Serial: COM1 | 4, 25 | 18 Rows, 75 Cols | Linux

(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 be 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
```

4.9.2 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-2.6.36/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: | |

Address: Room 1701,Block A2, 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



```
#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

4.9.3 User Button Test Program

| Program Description: | |
|----------------------|---|
| Source Code Location | /opt/FriendlyARM/mini6410/linux/linux-2.6.36/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: | |
| <pre>#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++; } } }</pre> | |



```
if (count_of_changed_key) {  
    printf("\n");  
}  
}  
close(buttons_fd);  
return 0;  
}
```

You can compile the program, download it and run

4.9.4 PWM Buzzer

| Program Description: | |
|-----------------------------------|---|
| Source Code Location | /opt/FriendlyARM/mini6410/linux/linux-2.6.36/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 terminal setting
if(tcsetattr(STDIN_FILENO,TCSANOW,&oldt) < 0) {
perror("restore the terminal 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

4.9.5 I2C-EEPROM Program

Program Description:

| | |
|-----------------------------------|---|
| Source Code Location | /opt/FriendlyARM/mini6410/linux/linux-2.6.36/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 *eeprom_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", eeprom_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

4.9.6 Serial Port Program

| Program Description: | |
|-----------------------------------|--|
| Source Code Location | /opt/FriendlyARM/mini6410/linux/linux-2.6.36/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 Friendly ARM to test serial ports. It is very similar to “minicom” in Linux and independent of hardware. Its source code can be applied in both any arm-linux platforms and PCs. **This program is developed by Friendly ARM and unauthorized usage of it is forbidden**

| Program: |
|---|
| <pre># 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);</pre> |



```
# 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));
```



```
}

#define 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

4.9.7 UDP Program

Program Description:

| | |
|-----------------------------------|---|
| Source Code Location | /opt/FriendlyARM/mini6410/linux/linux-2.6.36/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[BUFSIZE+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, BUFSIZE, 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, BUFSIZE, 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:

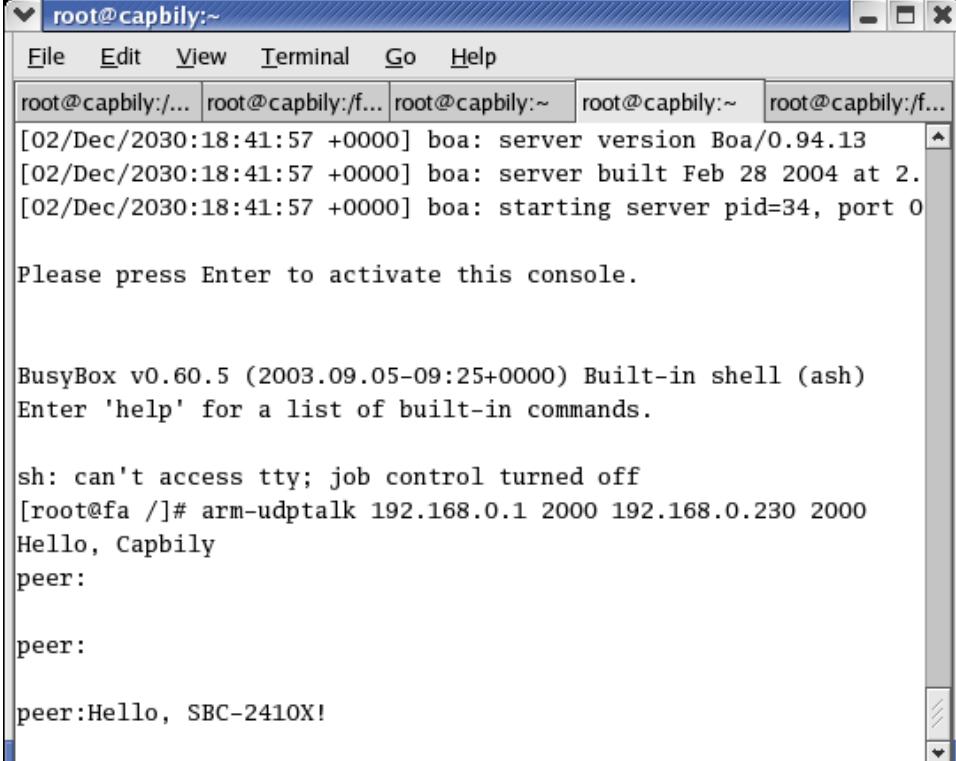


```
root@capbily:/friendly-arm/examples/udptalk
File Edit View Terminal Go Help
root@capbily:/... root@capbily:/f... root@capbily:~ root@capbily:~ root@capbily:/f...
[root@capbily udptalk]# ./x86-udptalk
./x86-udptalk <dest IP address> <dest port> <source IP address>
<source port>
[root@capbily udptalk]# ./x86-udptalk 192.168.0.230 2000 192.168
.0.1 2000

peer:

peer:Hello, Capbily
Hello, SBC-2410X!
peer:
```

x86-udptalk running on host



```
root@capbily:~
File Edit View Terminal Go Help
root@capbily:/... root@capbily:/f... root@capbily:~ root@capbily:~ root@capbily:/f...
[02/Dec/2030:18:41:57 +0000] boa: server version Boa/0.94.13
[02/Dec/2030:18:41:57 +0000] boa: server built Feb 28 2004 at 2.
[02/Dec/2030:18:41:57 +0000] boa: starting server pid=34, port 0

Please press Enter to activate this console.

BusyBox v0.60.5 (2003.09.05-09:25+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

sh: can't access tty; job control turned off
[root@fa /]# arm-udptalk 192.168.0.1 2000 192.168.0.230 2000
Hello, Capbily
peer:

peer:
```

arm-udptalk running on board

4.9.8 Application of 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: | |
| <pre>#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; }</pre> | |
| 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 *~ | |

4.9.9 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 *
```

4.9.10 Pipe Programming – Manipulate LED via WEB

| | |
|---|---|
| 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;  
            }  
            sprintf(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
```

4.9.11 “Hello, World” C++ Program

| | |
|----------------------|--|
| 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:";  
}
```

4.10 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

4.10.1 Hello Module

| Program Description: | |
|--|---|
| Source Code Location | |
| Driver | |
| Device Type | |
| Device Name | |
| Test Program Source Code Location | /opt/FriendlyARM/mini6410/linux/linux-2.6.36/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: | |
| <pre>#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);</pre> | |



```
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

```
root@tom:/opt/FriendlyARM/mini6410/linux-2.6.28.6/drivers/char
File Edit View Terminal Tabs Help
default y
help
    Say Y here if you want to support the /dev/kmem device. The
    /dev/kmem device is rarely used, but can be used for certain
    kind of kernel debugging operations.
    When in doubt, say "N".

config LEDS_MINI6410
    tristate "LED Support for Mini6410 GPIO LEDs"
    depends on CPU_S3C6410
    default y
    help
        This option enables support for LEDs connected to GPIO lines
        on Mini6410 boards.

config MINI6410_HELLO_MODULE
    tristate "Mini6410 module sample"
    depends on CPU_S3C6410
    help
        Mini6410 module sample.

config MINI6410_BUTTONS
    tristate "Buttons driver for FriendlyARM Mini6410 development boards"
    depends on CPU_S3C6410
    default y
    help
        this is buttons driver for FriendlyARM Mini6410 development boards

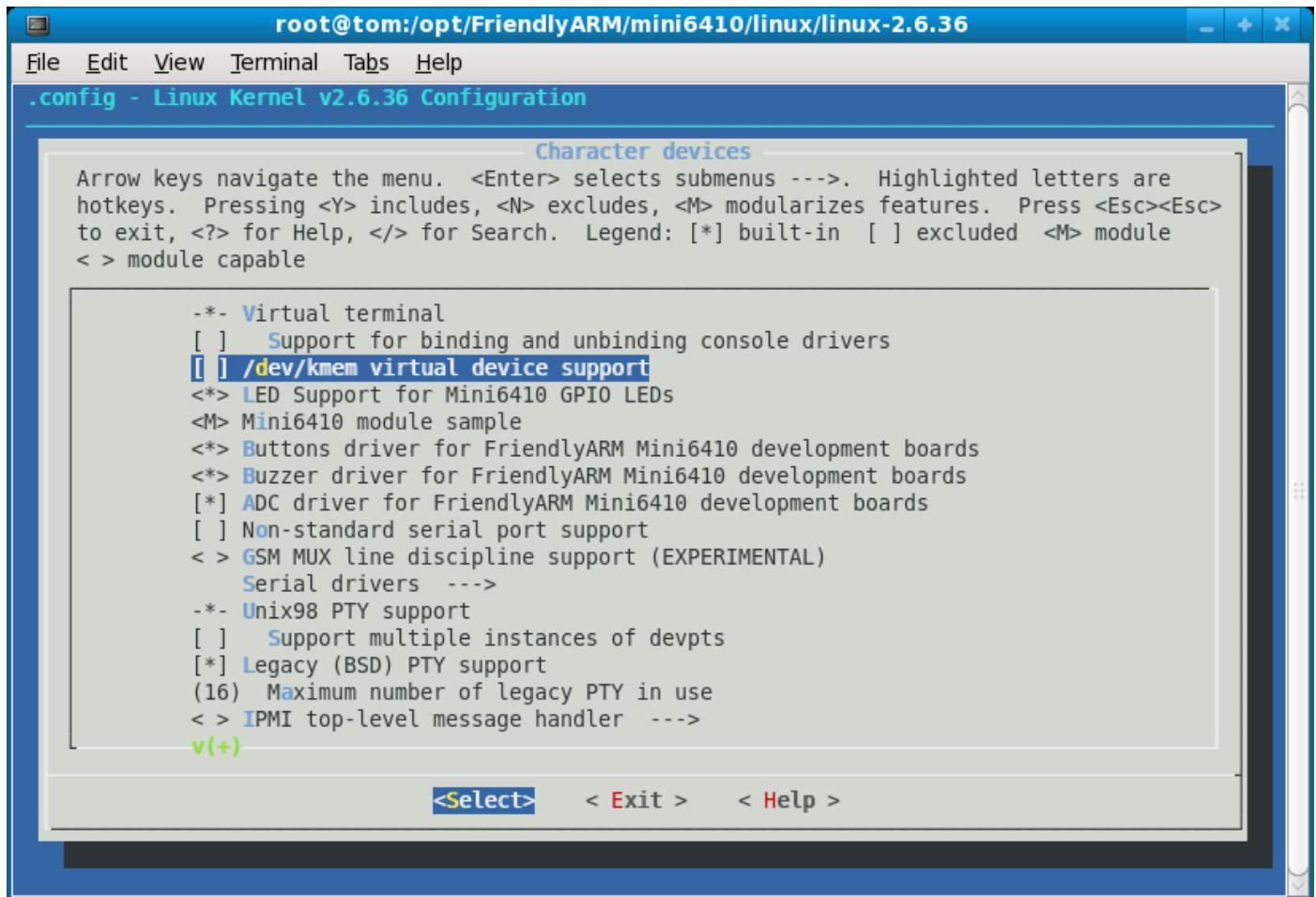
config MINI6410_BUZZER
    tristate "Buzzer driver for FriendlyARM Mini6410 development boards"
    depends on CPU_S3C6410
    default y
```

107,0-1

8%

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 source code will be compiled as a module. Press the space key again it will be marked "<*>". This means it will be compiled into the kernel. Here we chose "<M>"



Step2: the previous step still cannot include it into the kernel when compiling. You need to link the 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

```
root@tom:/opt/FriendlyARM/mini6410/linux/linux-2.6.36
File Edit View Terminal Tabs Help
obj-$(CONFIG_AGP)          += agp/
obj-$(CONFIG_PCMCIA)        += pcmcia/
obj-$(CONFIG_IPMI_HANDLER)  += ipmi/

obj-$(CONFIG_HANGCHECK_TIMER)  += hangcheck-timer.o
obj-$(CONFIG_TCG_TPM)        += tpm/

obj-$(CONFIG_PS3_FLASH)      += ps3flash.o
obj-$(CONFIG_RAMOOPS)        += ramoops.o

obj-$(CONFIG_JS_RTC)         += js-rtc.o
js-rtc-y = rtc.o

obj-$(CONFIG_MINI6410_LEDS)   += mini6410_leds.o
obj-$(CONFIG_MINI6410_HELLO_MODULE)  += mini6410_hello_module.o
obj-$(CONFIG_MINI6410_BUTTONS)  += mini6410_buttons.o
obj-$(CONFIG_MINI6410_BUZZER)   += mini6410_pwm.o
obj-$(CONFIG_MINI6410_ADC)     += mini6410_adc.o

# Files generated that shall be removed upon make clean
clean-files := consolemap_deftbl.c defkeymap.c

quiet_cmd_conmk = CONMK  $@
cmd_conmk = scripts/conmakehash $< > $@

$(obj)/consolemap_deftbl.c: $(src)/$(FONTPMAPFILE)
    $(call cmd,conmk)

$(obj)/defkeymap.o: $(obj)/defkeymap.c
"drivers/char/Makefile" 147L, 4806C
119,1           88%
```

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.

```
root@tom:/opt/FriendlyARM/mini6410/linux/linux-2.6.36
File Edit View Terminal Tabs Help
[root@tom linux-2.6.36]# make modules
CHK include/linux/version.h
CHK include/generated/utsrelease.h
make[1]: `include/generated/mach-types.h' is up to date.
CALL scripts/checksyscalls.sh
CC [M] drivers/char/mini6410_hello_module.o
Building modules, stage 2.
MODPOST 20 modules
LD [M] drivers/char/mini6410_hello_module.ko
[root@tom linux-2.6.36]#
```

(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.isapnmap   modules.usbmap
modules.ccmmap  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]#
```

4.10.2 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-2.6.36/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: | |
| <pre>#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></pre> | |



```
#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.");
```

4.10.3 User Button Driver

| |
|----------------------|
| Program Description: |
|----------------------|

| |
|----------------------|
| Source Code Location |
|----------------------|

| |
|---|
| /opt/FriendlyARM/mini6410/linux/linux-2.6.36/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.");

```

4.11 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 the whole qtopia platform and its utilities. You can start them by commanding “**run**”. The compiling scripts for x86 and arm are a little bit different.

4.11.1 Uncompress and Install Source Code

Please refer to 4.4.1

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
```

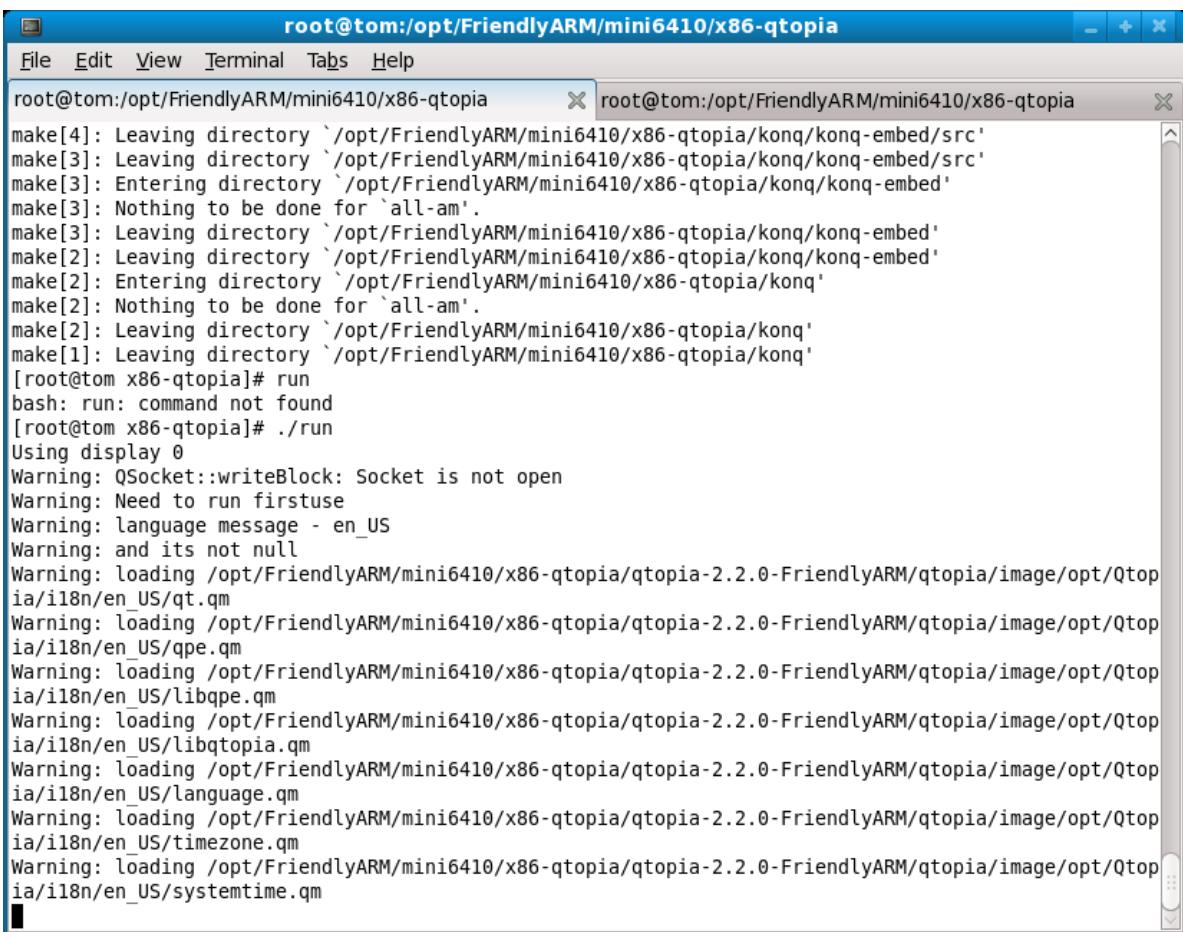
```
#!/build-all (this process takes about 30 minutes)
```

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.

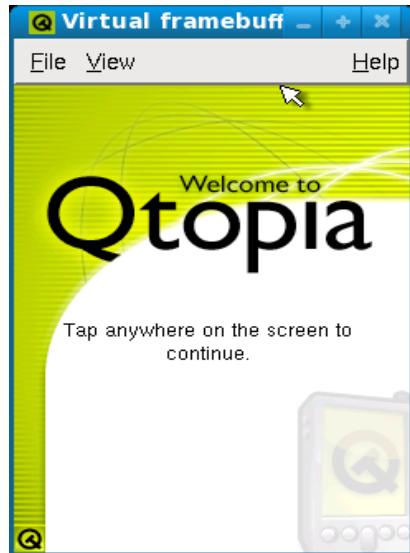
To run your qtopia you can type the command below:

```
#!/run
```

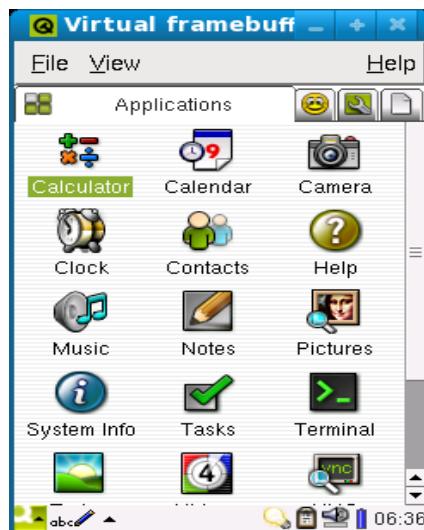
You will see the following screen



```
root@tom:/opt/FriendlyARM/mini6410/x86-qtopia
File Edit View Terminal Tabs Help
root@tom:/opt/FriendlyARM/mini6410/x86-qtopia  X root@tom:/opt/FriendlyARM/mini6410/x86-qtopia  X
make[4]: Leaving directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq/konq-embed/src'
make[3]: Leaving directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq/konq-embed/src'
make[3]: Entering directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq/konq-embed'
make[3]: Nothing to be done for `all-am'.
make[3]: Leaving directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq/konq-embed'
make[2]: Leaving directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq/konq-embed'
make[2]: Entering directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq'
make[2]: Nothing to be done for `all-am'.
make[2]: Leaving directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq'
make[1]: Leaving directory `/opt/FriendlyARM/mini6410/x86-qtopia/konq'
[root@tom x86-qtopia]# run
bash: run: command not found
[root@tom x86-qtopia]# ./run
Using display 0
Warning: QSocket::writeBlock: Socket is not open
Warning: Need to run firstuse
Warning: language message - en_US
Warning: and its not null
Warning: loading /opt/FriendlyARM/mini6410/x86-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/image/opt/Qtopia/i18n/en_US/qt.qm
Warning: loading /opt/FriendlyARM/mini6410/x86-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/image/opt/Qtopia/i18n/en_US/qpe.qm
Warning: loading /opt/FriendlyARM/mini6410/x86-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/image/opt/Qtopia/i18n/en_US/libqpe.qm
Warning: loading /opt/FriendlyARM/mini6410/x86-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/image/opt/Qtopia/i18n/en_US/libqtopia.qm
Warning: loading /opt/FriendlyARM/mini6410/x86-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/image/opt/Qtopia/i18n/en_US/language.qm
Warning: loading /opt/FriendlyARM/mini6410/x86-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/image/opt/Qtopia/i18n/en_US/timezone.qm
Warning: loading /opt/FriendlyARM/mini6410/x86-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/image/opt/Qtopia/i18n/en_US/systemtime.qm
```



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



4.11.3 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)
```

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

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

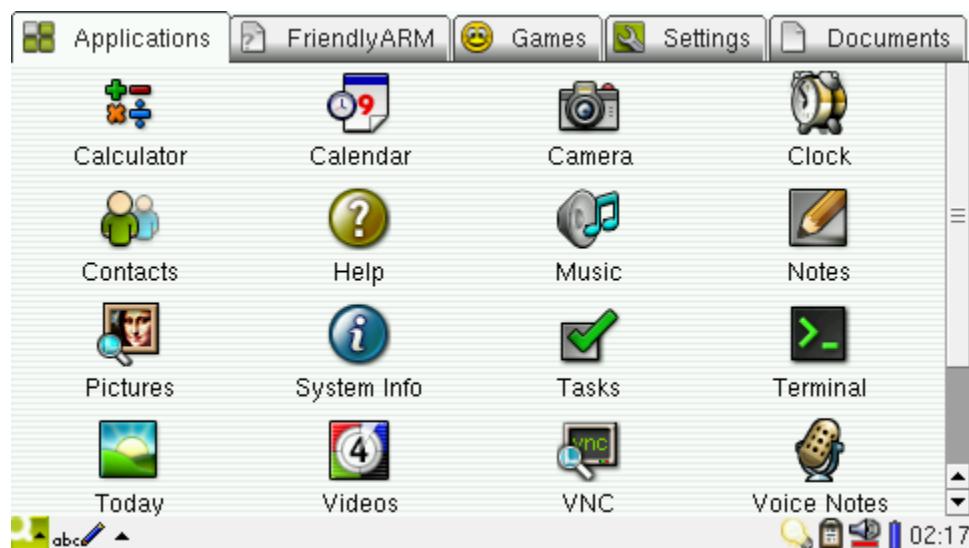
To remove your old Qtopia system you just need to delete all the files under “/opt”. Then you can uncompress your target-qtopia-konq.tgz to the board’s root directory via a flash drive. In our example we had it under /home/plg. Please run the command below:

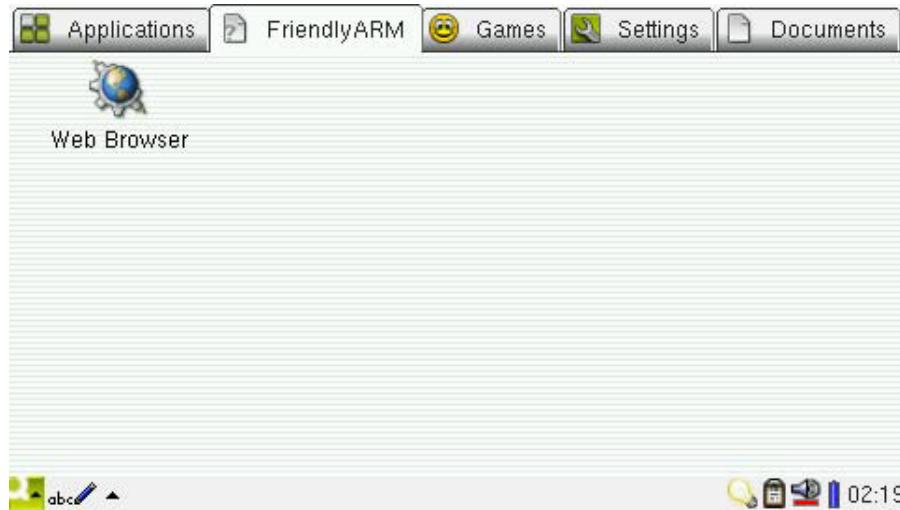
```
#tar xvzf /home/plg/target-qtopia-konq.tgz -C /
```

“C” means “Change” and “/” after “C” means it will be uncompressed to the root directory.

After you are done, reboot your board and you will see that all your GUI components are in English now and there is a browser under the “FriendlyARM” tag. This is your own Qtopia.

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





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

4.12 Compile QtE-4.8.5

4.12.1 Uncompress and Install Source Code

Please refer to 4.4.1

4.12.2 Compile and Run QtE-4.8.5 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-4.8.5. Please enter the source code directory and type the following command:

```
#cd /opt/FriendlyARM/mini6410/linux/arm-qte-4.8.5
```

```
#!/build.sh
```

The build process takes a while. And after it is done, please run the mktarget script and a

target-qte-4.8.5-to-devboard.tgz and a **target-qte-4.8.5-to-hostpc.tgz** will be generated.

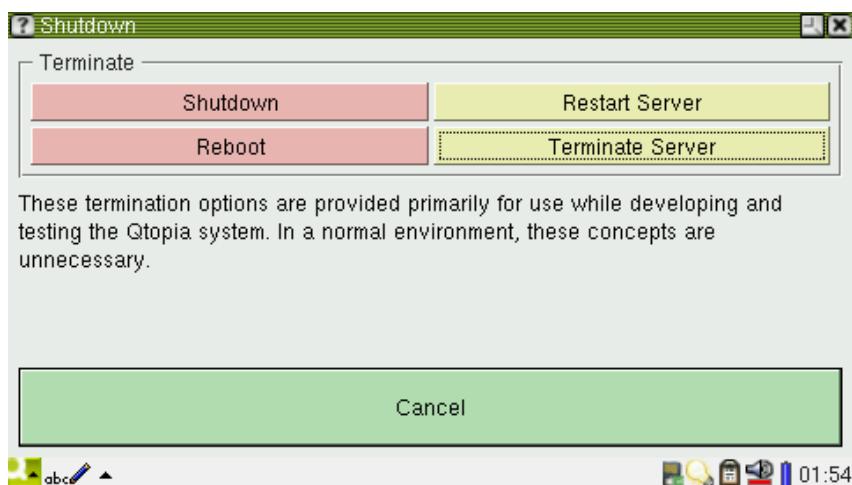
The **target-qte-4.8.5-to-devboard.tgz** is for our 6410 board. The **target-qte-4.8.5-to-hostpc.tgz** is for PCs which users can develop applications based on after they install it on a PC. What we will show here is the **target-qte-4.8.5-to-devboard.tgz**.

Please follow the command below:

```
#tar xvzf target-qte-4.8.5-to-devboard.tgz -C /
```

A Trolltech directory will be generated under “/usr/local/”, which includes all needed libraries and executables.

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



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

qtopia-2.2.0 and run “qt4”



Chapter 5 Explore WinCE6

5.1 Get Started with CE6

The image file for WindowsCE6 is under “\images\WindowsCE6”.Please follow the steps described in our previous chapters to burn a WinCE 6 image into the board (we used NK_T43-i.bin which is for a 4.3”LCD and supports accurate touching). After your burning is done please toggle the S2 switch to the Nand Flash and reboot the system.

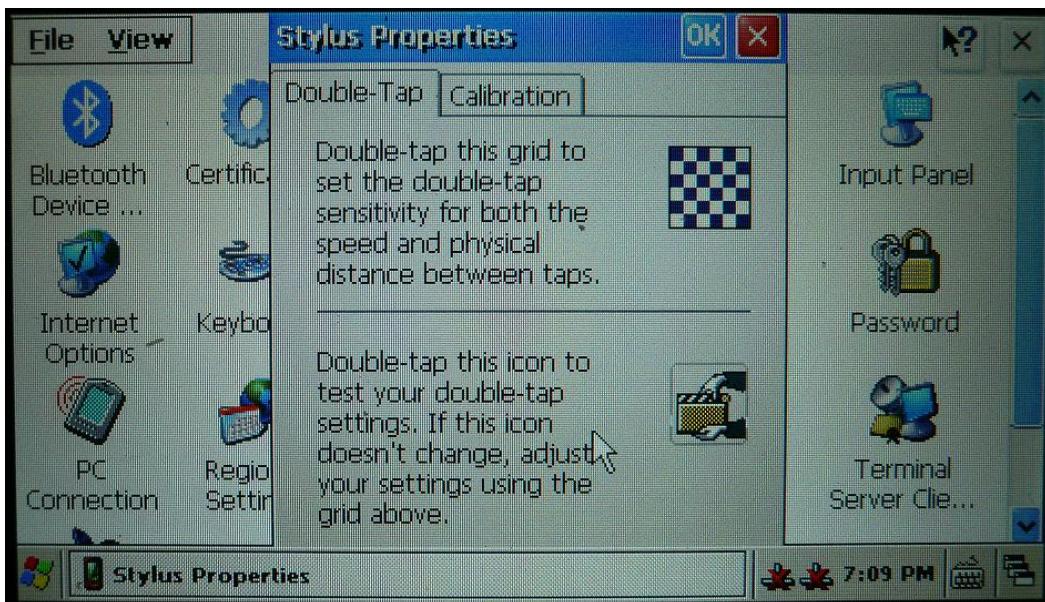


5.1.1 Calibrate Touch Screen

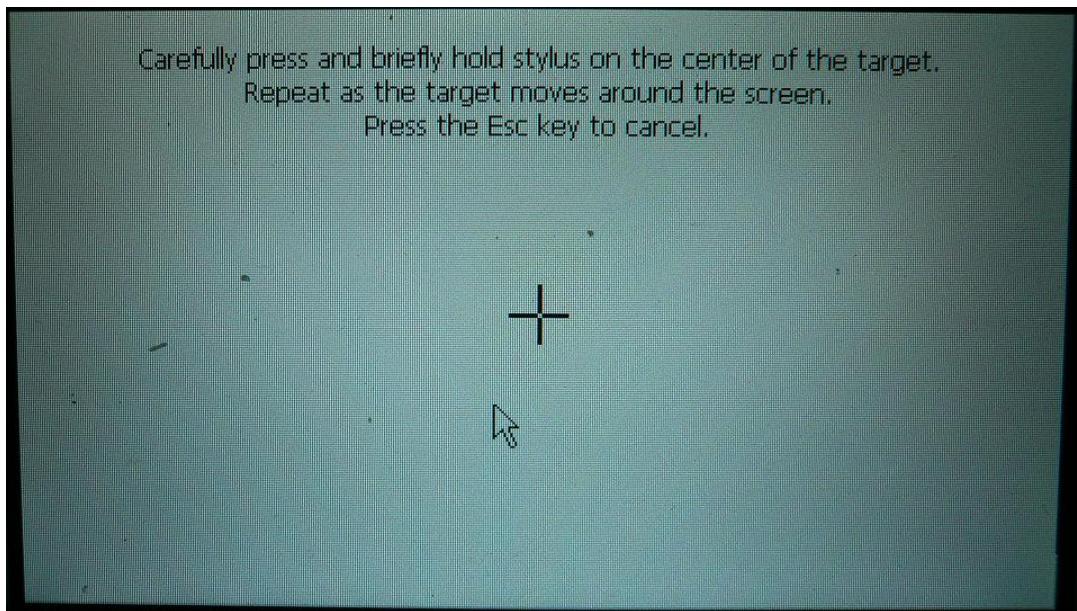
The default WinCE system's touch screen parameters are for NEC 4.3”LCD. Other LCD systems may require different settings therefore users need to re-calibrate the screen. Below

are the steps:

Step 1: connect a USB mouse to your board, go to “Start -> Settings -> Control Panel”, locate the “Stylus” icon, double click to open its property window and click on “Calibrate -> ReCalibrate”



Follow the system's prompt to start calibration. After you are done you will see the following screen. Click on any position you will return to the property window. Please click on “OK” to save and exit.



If you want to save the setting, you can go to “Start -> Suspend” and reboot.



5.1.2 Screen Rotation

Go to “FriendlyARM” and click on the “Rotate” icon you will see the following effect.



5.1.3 Check System Info

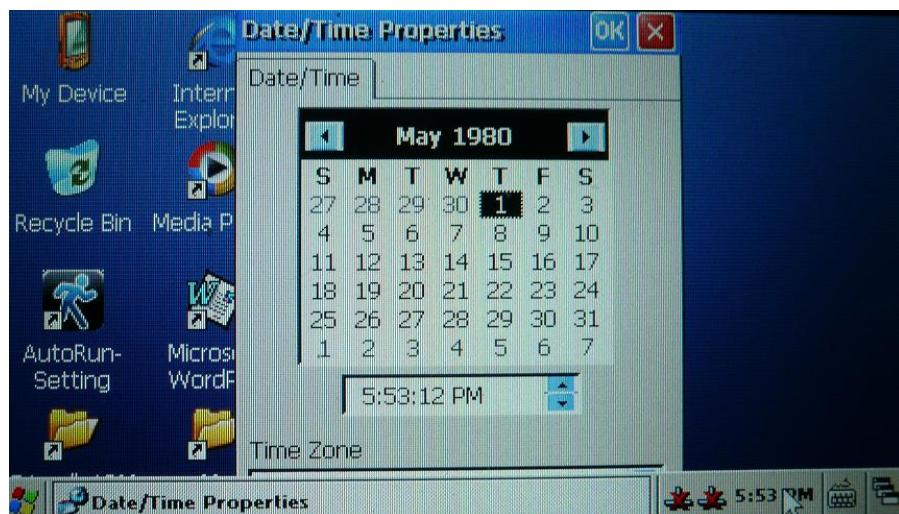
Go to “start -> Settings -> Control Panel -> System” and you can check your system info.

Or you can right click on “My Device -> Property”.



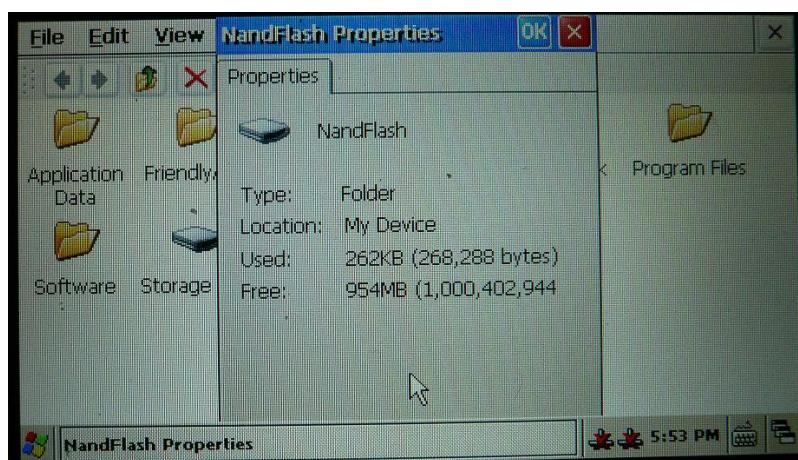
5.1.4 Set Time Zone and Date

Click on the time area on the bottom right corner, the time setting interface will pop up. You can just follow its prompts to set time and date. Click on “OK” to save and return and your settings will be saved



5.1.5 User Data

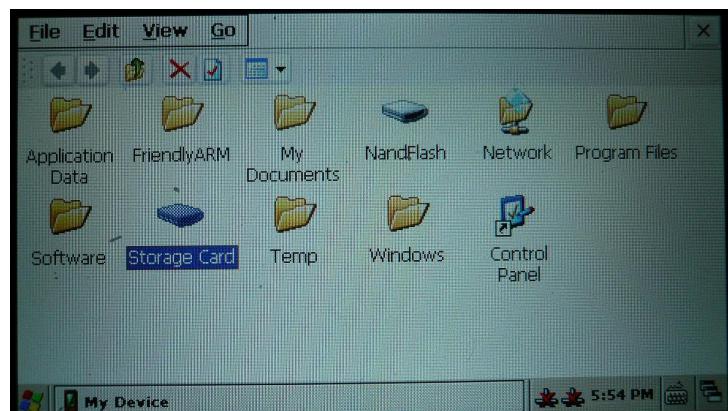
Open “My Device”, you will see a “NandFlash” icon. Users can save data in it and will not lose it even after the system is powered off.



5.1.6 Mount Flash Drive/SD Card

In WinCE we can use use flash drives. After WinCE is booted insert a USB flash drive into the host socket, seconds later the drive will be automatically mounted. Double click on “My Device” and you will see it. Then you can enter it and operate your files.

You can do it the same way for SD cards. Insert an SD card into the SD card socket and you will see it listed as “Storage Card”. Enter it and you will be able to operate your files.



5.1.7 Play MP3

Users can use WinCE’s MediaPlayer to play mp3



5.1.8 LED Test

Go to “Friendly ARM”, click on “LED-Test” you will see the following dialog and you can manipulate LEDs by clicking on the buttons on it



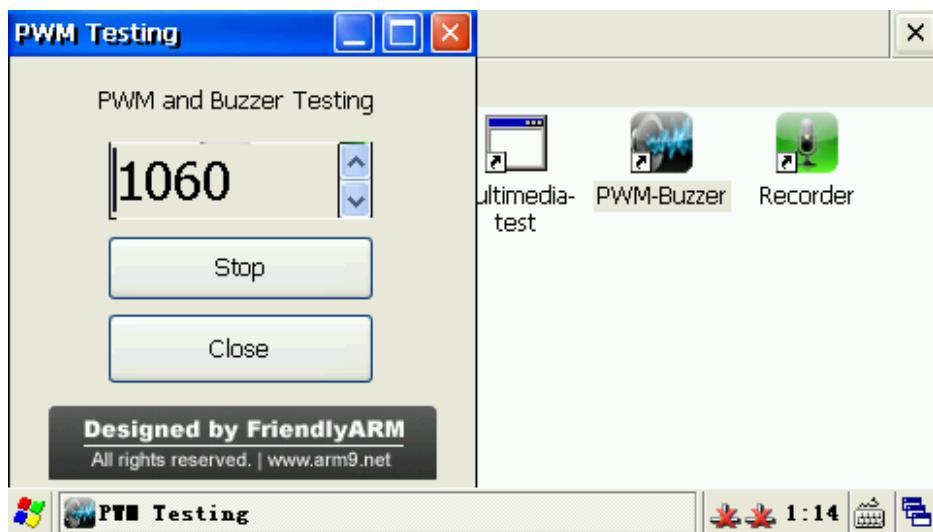
5.1.9 User Button Test

Go to “FriendlyARM”, click on “Buttons” you will see the following dialog. Clicking on the buttons you will observe their color changes



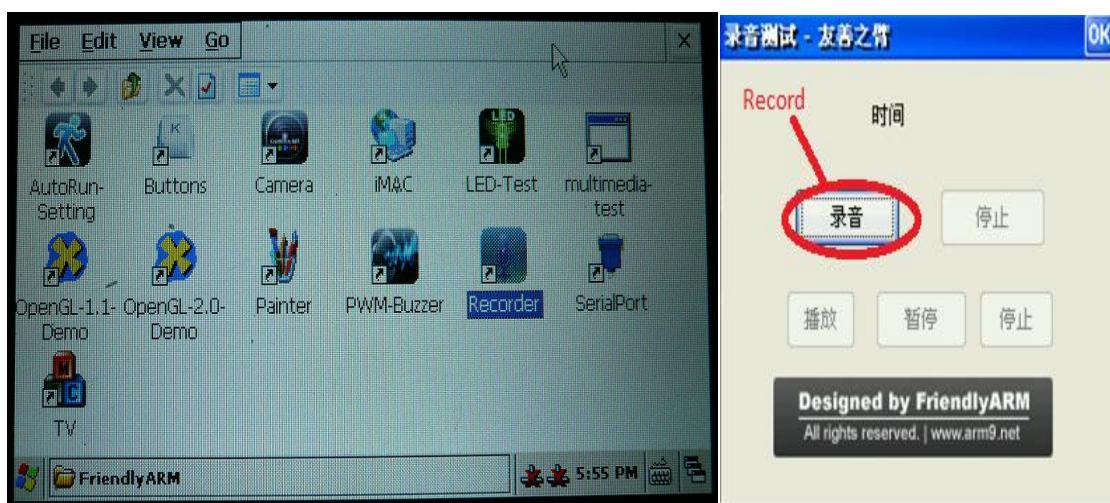
5.1.10 PWM Buzzer Test

Go to “FriendlyARM”, click on “PWM-Buzzer” you will see the following dialog. Click on “Start” you can test its beeping. Click on “Stop” you can stop it.



5.1.11 Audio Recording

Go to “FriendlyARM”, click on “Recorder”.



Click on “Record” to begin recording. Now if you speak to the microphone on the board your voice will be recorded. Click on “Stop” to stop recording



Click on “Play” it will loop the audio you just recorded.

Note: this utility doesn't save the recorded audio file.

5.1.12 Serial Port Assistant

Note: the Mini6410-1405 BSP includes drivers for three standard serial ports: COM2, 3 and

4. To test these three ports you need our extension board. Please hook up your extension board as follows:





Go to “Friendly ARM” and click on “SerialPort”, you will see the following dialog



Click on “Settings”, select COM2, set its bit rate to 115200, click on “OK” to save it.

Meanwhile connect your extension board(COM2) to PC, set up your PC’s corresponding COM.

In the main window, click on the “Open” button, (the button’s title will change to “Close”), type some characters in the edit area and click on the “Send” button. You will see the characters you typed received in your PC.



In the Serial Port Assistant’s main window, click on the “Receive” button (its title will

change to “Not Receive”), type some characters in the edit area of your PC’s serial port window, you will see the characters you typed received in the Assistant’s main window



We can follow this procedure to test COM3 and COM4 too

5.1.13 “Hard Decoding” Player

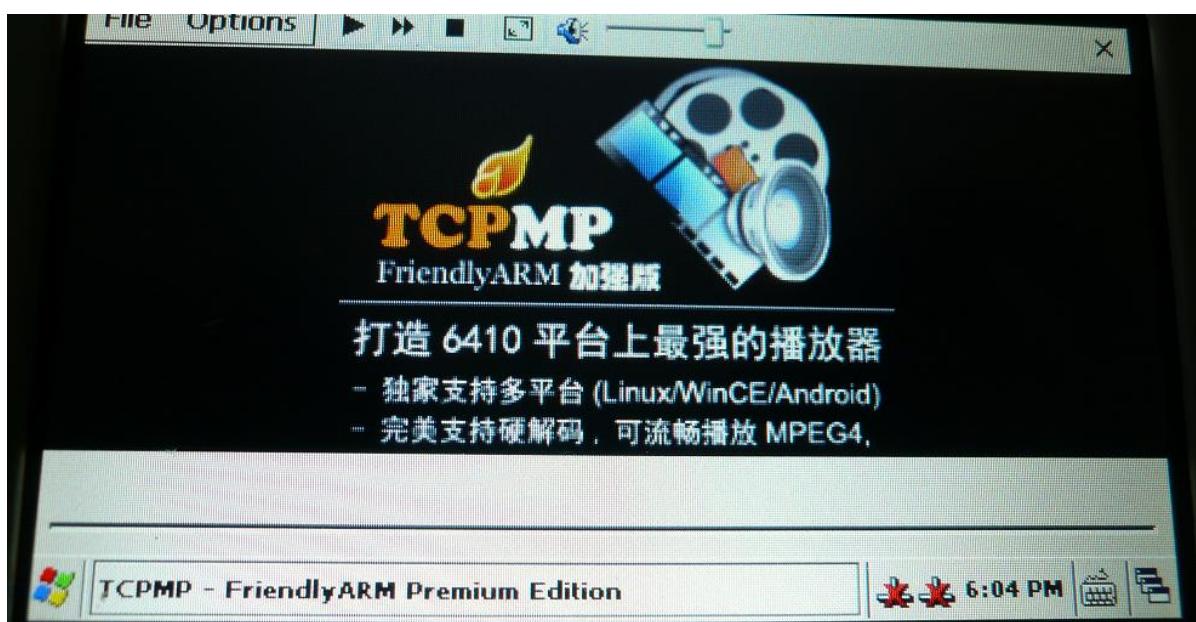
We developed a “hard decoding” player in our Mini6410-1405 system that is able to play 720x480 30fps or 720x576 25fps Mpeg4, H264, H263 videos. (We are the first one to implement this feature in Linux in China). This player has the following features:

- it can automatically recognize Mpeg4, H264 and H263 files and switch to hard decoding play. That is to say if your Mini6410-1405 platform doesn’t support hard decoding it can play these files too although it may not play that smoothly
- elegant playing
- utilizes the DirectDraw technology

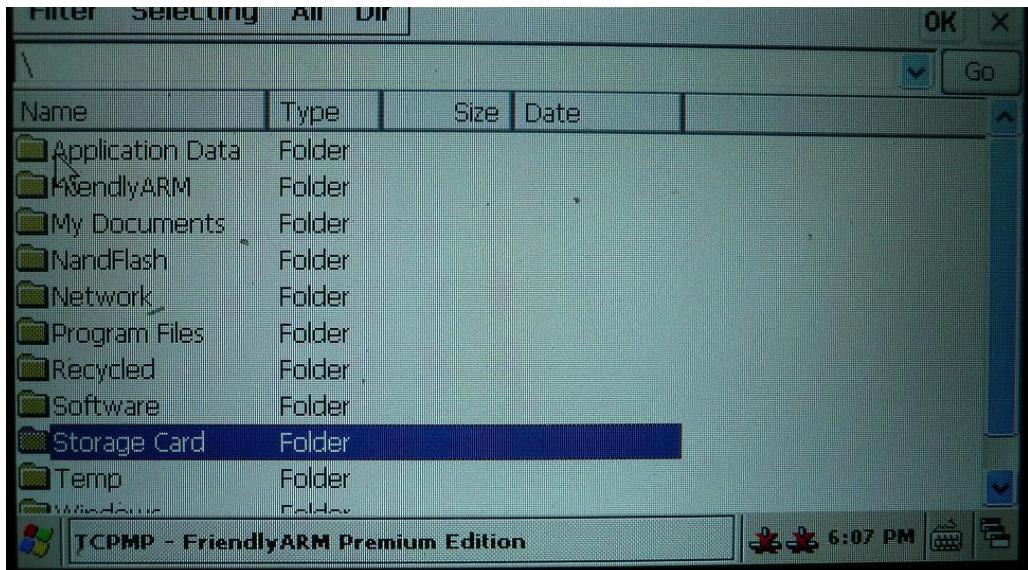
This player by default has been integrated in WinCE. We provide two test videos which are under “\test videos”. You can copy them (one is a H.264 file and the other is a MPEG4 file) to your SD card.

Click on the “player” icon on WinCE’s desktop.

Note: when you launch this utility in a newly installed WinCE it may not run that fast.



Go to “File -> Open File” to select your video file



Double click on your file and it will be played



Note: the maximum hard decoding resolution the Mini6410-1405 system can support is 720x480 therefore when you play a video in a 7" LCD in full screen it may not go very smoothly. You can go to “Options->View->Zoom->100%” and “Options->View->Pixel Aspect Ratio” and click on full screen to fix this issue. For other LCDs you can make adjustments accordingly.

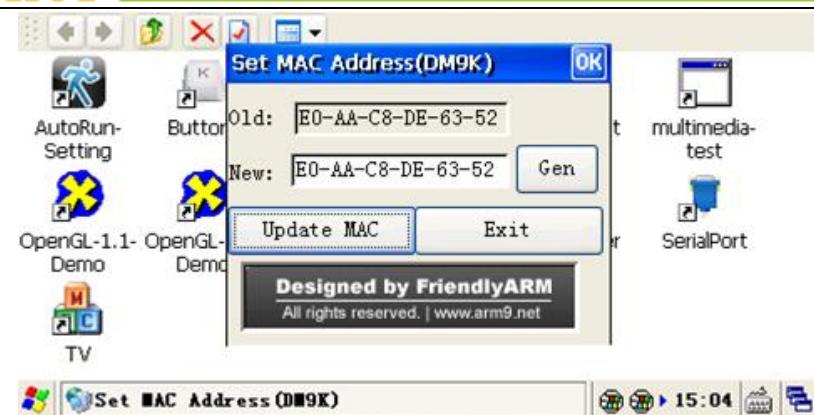
5.1.14 Set MAC Address

The integrated DM9000 network card doesn't come with a MAC address therefore we strongly recommend our users to set it prior to connecting to the internet after burning an image into the board.

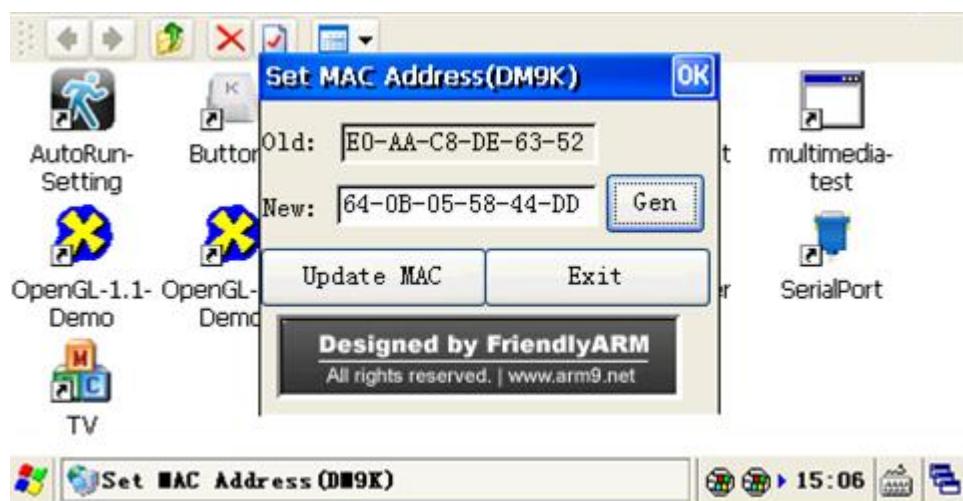
After you set up your MAC address it will be written into the registry and will be permanent unless you reinstall your system or update it. Click on the "iMAC" icon to start the utility.



On the MAC Address setting dialog, "Old" shows the current MAC. You can type your new in "New" or click on the "Gen" button to generate a random MAC which in general achieves better results:



The following screens shows a MAC generated by “Gen”



Click on “Update MAC” to save your MAC into the registry and reboot your system



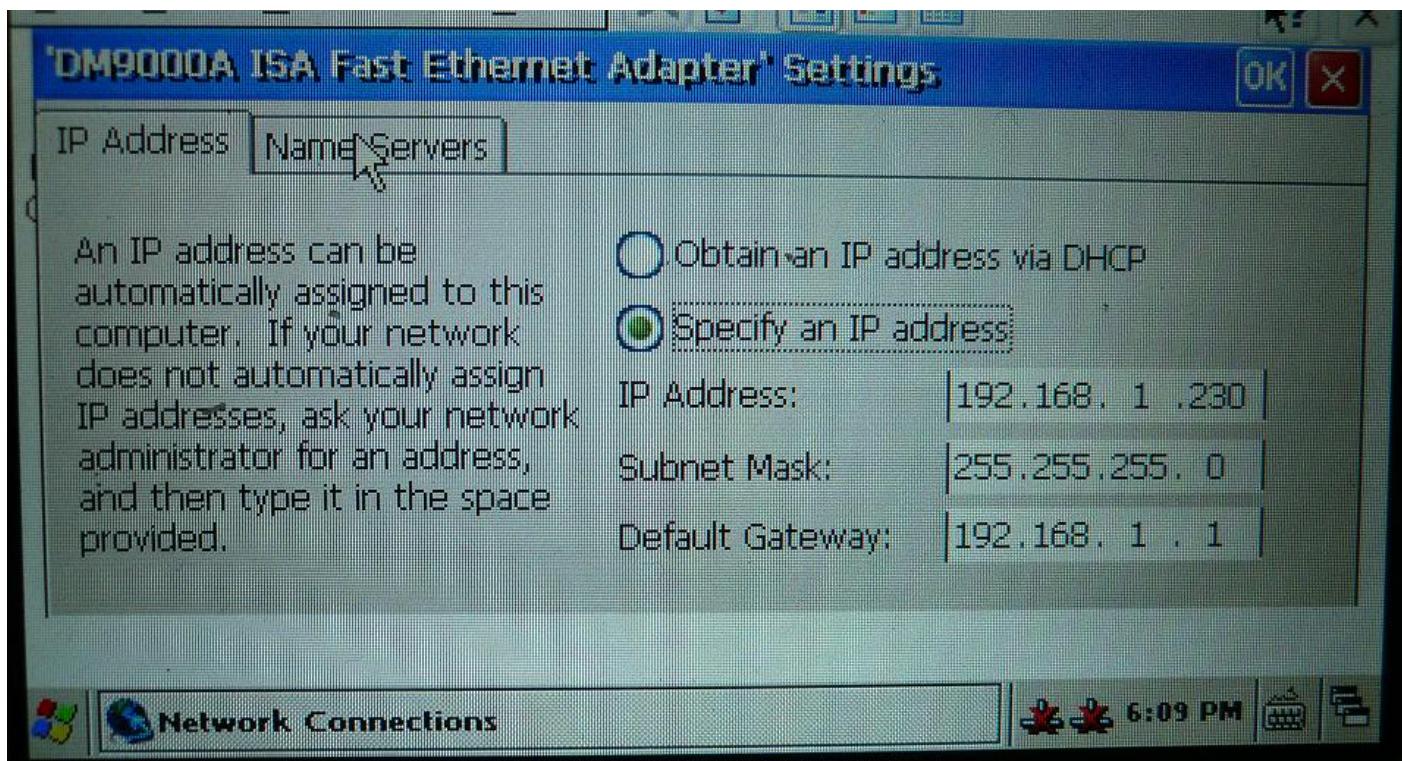
After system is rebooted you MAC will take into effect

5.1.15 Configure Ethernet

Before you can browse the internet you need to set up your IP, gateway and DNS properly. Please go to “Start -> Settings->Control Panel”, launch the network setting utility and locate your DM9CE1.



Double click on the DM9CE1 icon, you will see the following dialog in which you can make your network configurations.

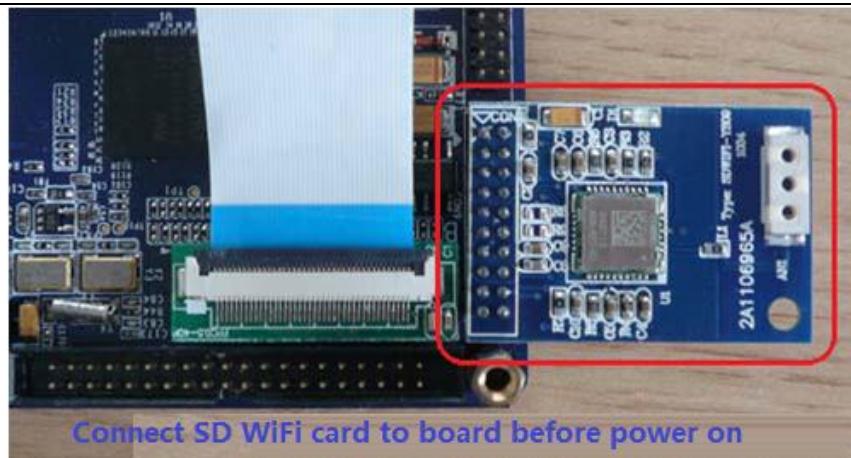


After setting up your network properly you can try it now



5.1.16 SD WiFi

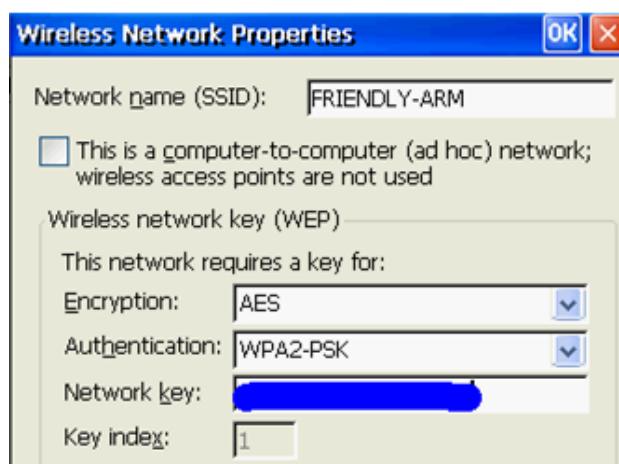
Before boot your system, please connect an SD-WiFi to your board's SDIO (CON9).



Boot the system and you will see the SD WiFi's green led is flashing and it is searching for a nearby network. The following dialog will pop up if it does find one

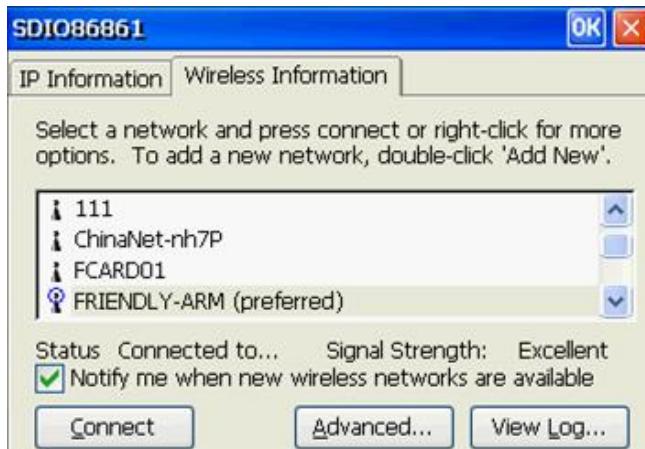


Select a network, click on “Connect” and type the required information to connect



Click on “OK” on the upper right of the dialog you will see the following dialog if your

connection is successful.



5.1.17 USB WiFi

The Mini6410-1405 system (WinCE) by default integrates the driver for Ralink RT2070/RT3070 USB WiFi network card. This card is plug and play. Its configuration GUI is the same as the SD WiFi GUI. The following screenshot is the GUI



The other configurations are the same as the SD WiFi configurations

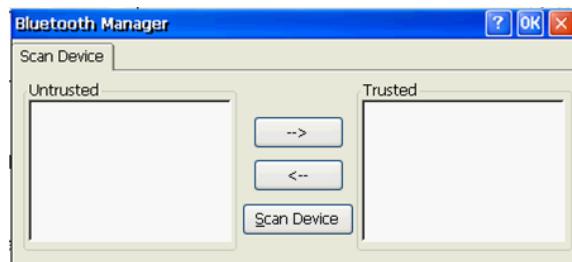
5.1.18 USB Bluetooth

The Mini6410-1405 system(WinCE) by default integrates the USB bluetooth driver. This driver is not from the third party but comes with WinCE. You only need to configure it. In

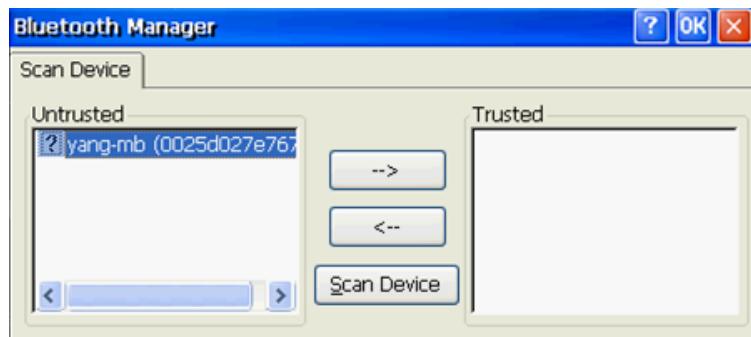
in our example project we have configured this option therefore you can plug and play your USB Bluetooth. One thing to remind users is that it doesn't guarantee to support all USB Bluetooth modules

Below are the steps to follow:

Hook up your bluetooth module to your board and go to "Control Panel->Bluetooth Devices"



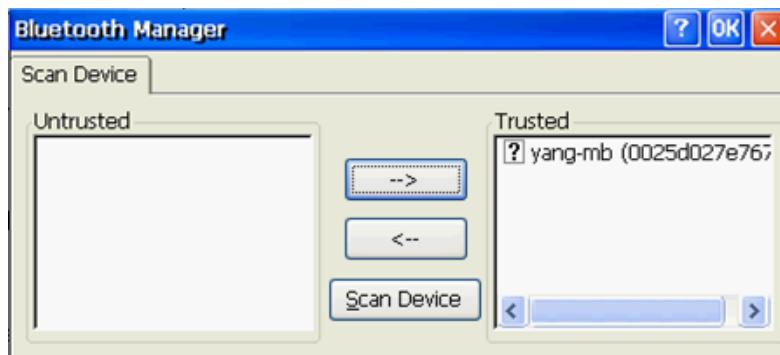
Click on "Scan Device" to search for nearby Bluetooth devices. In our example it found a cell phone



Click on "→" to add your device to "Trusted" and you will see the following dialog



Click on “Yes” a verification dialog will pop up, you can type several digits randomly such as “111” and you will have a dialog on your phone, type the same digits you did and your device will be added to “Trusted”



This way a trusted channel between your board and your cell phone will be established. After you send a file from your cell phone you will see a dialog on your board

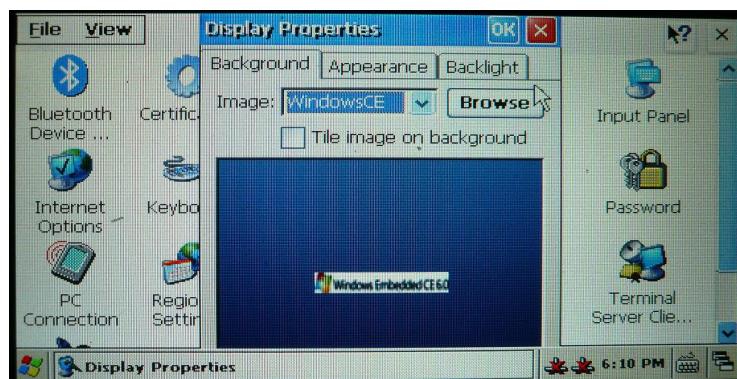


Click on “Yes” to accept it and the file will be saved under “Documents\DefaultInbox”

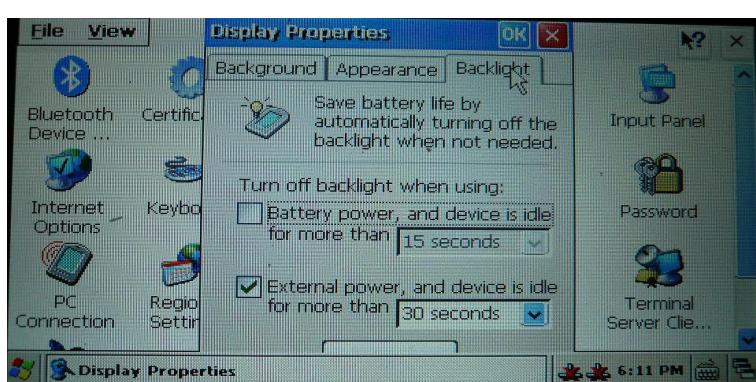
5.1.19 Backlight Control

If your system is preinstalled with WindowsCE6, you may notice that your LCD backlight will turn off if it doesn't accept any touch within 30 seconds. This is manipulated by the backlight control function. 4.3" and 7" LCDs that have the accurate touch function have the backlight control function. In WinCE the backlight control driver utilizes standard system interfaces and you can easily manipulate it via software.

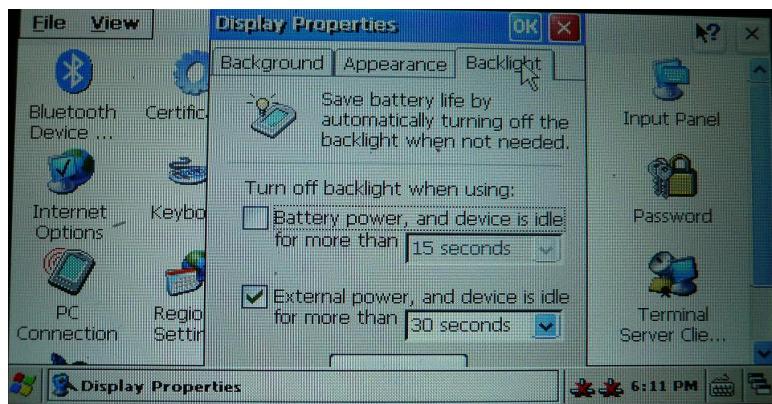
Please go to “Control Panel -> Display”



Click on the “backlight” tab you will be able to set the turn-off time. By default it is 30 seconds



Click on “Advanced”



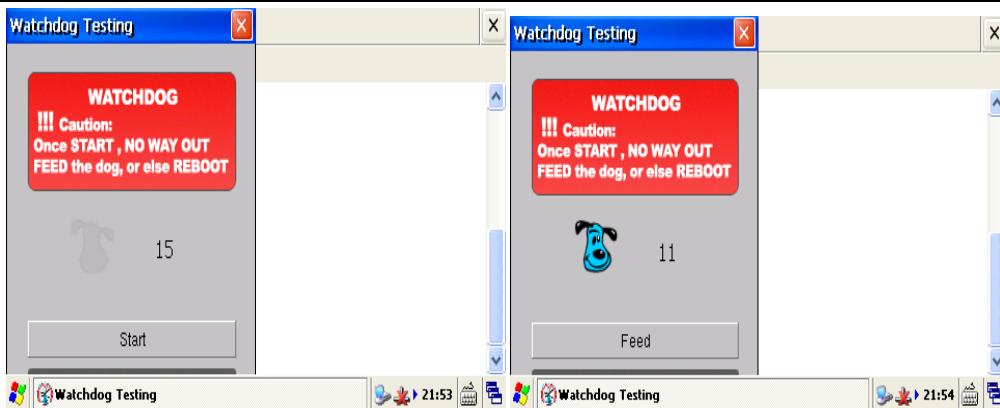
The backlight control window will pop up



You can slide the slider to adjust the backlight. Click on “close” to return to your previous interface

5.1.20 Watchdog

Please go to “FriendlyARM” and click on “Watchdog”



5.1.21 Synchronize with PC (Windows 7)

Note: in Windows 7, you don't need ActiveSync! Please make sure your PC can connect to the internet.

In Windows 7 the synchronization is implemented via “Windows Mobile Device” (Synchronization Center) which is similar to ActiveSync. Below is its interface



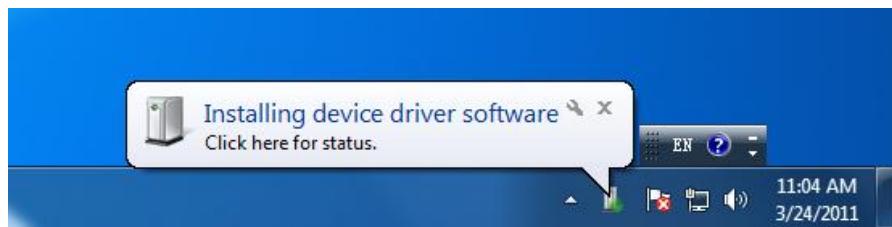
The “Synchronization Center” doesn't come with Windows 7 and needs to be installed.

Below are the steps to follow for Windows 7: (Note: if your board runs WinCE6 you still can

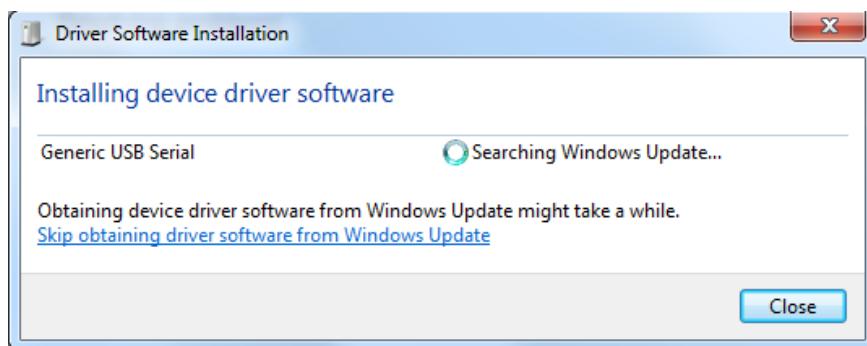
connect your board to your PC via ActiveSync.)

● Install Windows Mobile Device Center

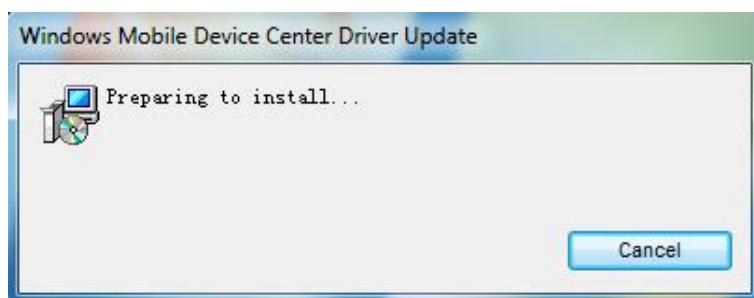
Power on your board that runs WinCE6, connect it to a PC that runs Windows 7 via USB you will see the following window



Soon you will see the following window



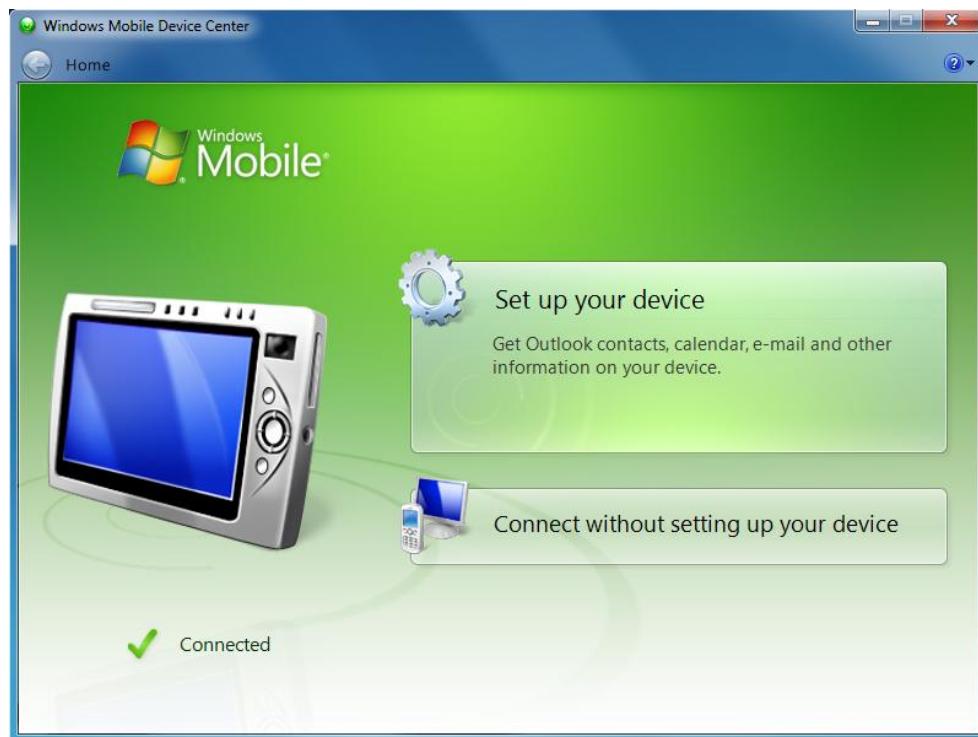
If your system is connected to the internet it will automatically download and install related software



After installation is done, you will see the following configuration dialog click on “Accept” on the license window



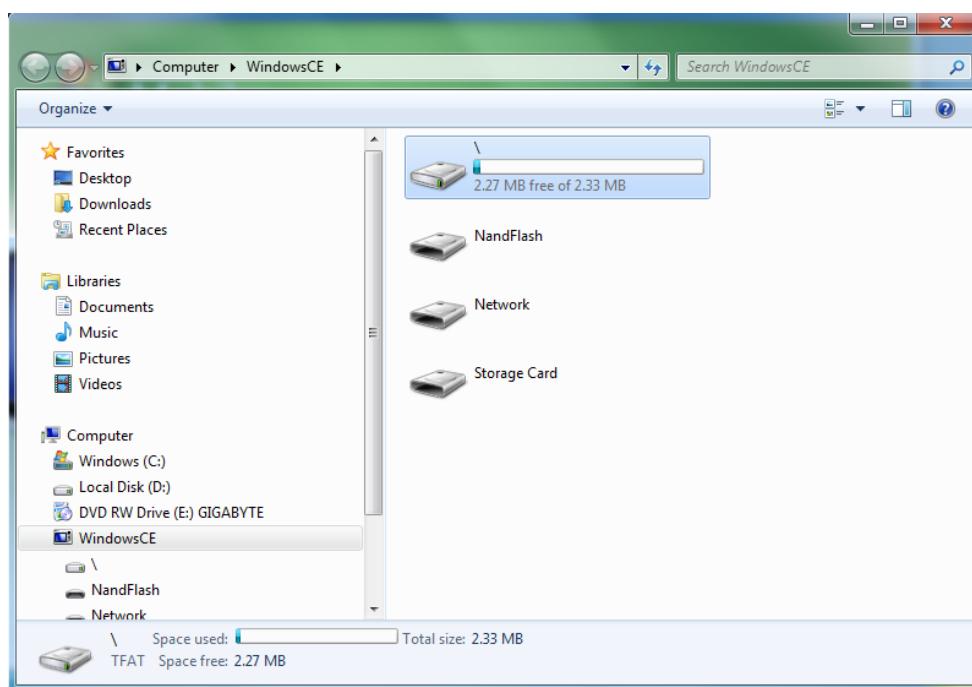
Your board is now connected to your PC



Click on “Connect”



Click on “File Management -> Browse” you will enter the board’s root directory. If your board has a SD card or USB drive connected it will be listed here too



Now you can manipulate your files between your PC and board

5.2 Set up Windows CE6 Development Environment

Note: the following software and installation steps are based on Windows 7 (flagship version).

We recommend users to copy the installation software to your hard disk to install



Installation of Windows CE 6.0 is complicated and has lots of requirements for hardware therefore we strongly suggest users exactly follow our installation steps:

Below are the basic system requirements

CPU: Intel Core Duo E8400

RAM: DDR2 4GB

Hard Disk: 500GB

Software List (We don't offer Windows Embedded 6.0 CE 6 installation files and users may need to download its trial version from Microsoft's website):

- ✓ Visual Studio 2005

(trial version download website:

http://download.microsoft.com/download/e/1/4/e1405d9e-47e3-404c-8b09-489437b27fb0/E_n_vs_2005_Pro_90_Trial.img)

- ✓ Visual Studio 2005 Service Pack 1(file name: VS80sp1-KB926601-X86-ENU.exe)

Download website:

<http://www.microsoft.com/downloads/details.aspx?familyid=bb4a75ab-e2d4-4c96-b39d-37ba6b5b1dc&displaylang=en>

- ✓ Visual Studio 2005 Service Pack 1 Update for Windows Vista (文件名:
VS80sp1-KB932232-X86-ENU.exe)

Download website:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=90E2942D-3AD1-4873-A2EE-4ACC0AACE5B6&displaylang=en>)

- ✓ Visual Studio 2005 Service Pack 1 ATL Security Update
(VS80sp1-KB971090-X86-INTL.exe)

Download website:

<http://www.microsoft.com/downloads/details.aspx?familyid=7C8729DC-06A2-4538-A90D-FF9464DC0197&displaylang=en>

- ✓ Windows Embedded CE 6.0

Trial version download website:

<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=7e286847-6e06-4a0c-8cac-ca7d4c09cb56>

- ✓ Windows Embedded CE 6.0 Platform Builder Service Pack 1

Download website:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=BF0DC0E3-8575-4860-A8E3-290ADF242678&displaylang=en>

- ✓ Windows Embedded CE 6.0 R2

Download website:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=F41FC7C1-F0F4-4FD6-9366-B61E0AB59565&displaylang=en>

- ✓ Windows Embedded CE 6.0 R3

Download website:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=BC247D88-DDB6-4D4A-A595-8EEE3556FE46&displaylang=ja&displaylang=en>

- ✓ Tencent QQ (third part software)

Download website:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=527042f7-bb5b-4831-a6ad-5081808824ec&displaylang=en>

- ✓ WesttekFileViewers6.exe(office file manager, third part software)

Download website:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=d2fd14eb-7d5c-428b-951c-343f910047c1&displaylang=en>

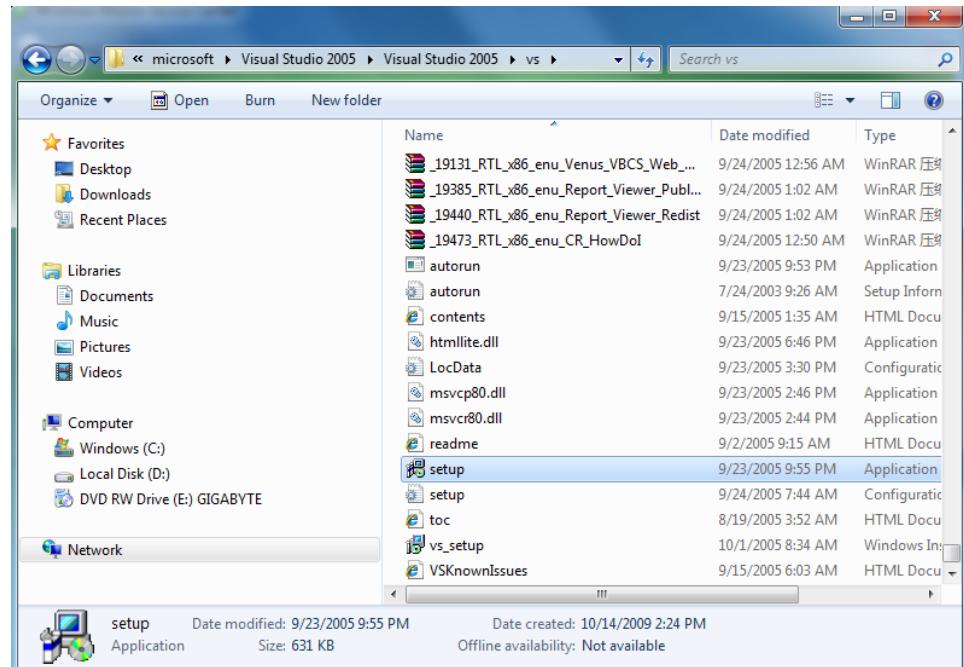
The above list also indicates the order of installing these software components: Visual Studio 2005 and its patches first, then Windows CE 6.0 and its patches and finally the third party software

Note: the Platform Builder for Windows CE 6.0 is different from its previous versions such as Windows CE 5.0/4.2. It is not a standalone software application but a plugin of VS2005 and therefore depends on VS2005. You need to install VS2005 first. All the configurations and compilation described below are with VS2005.

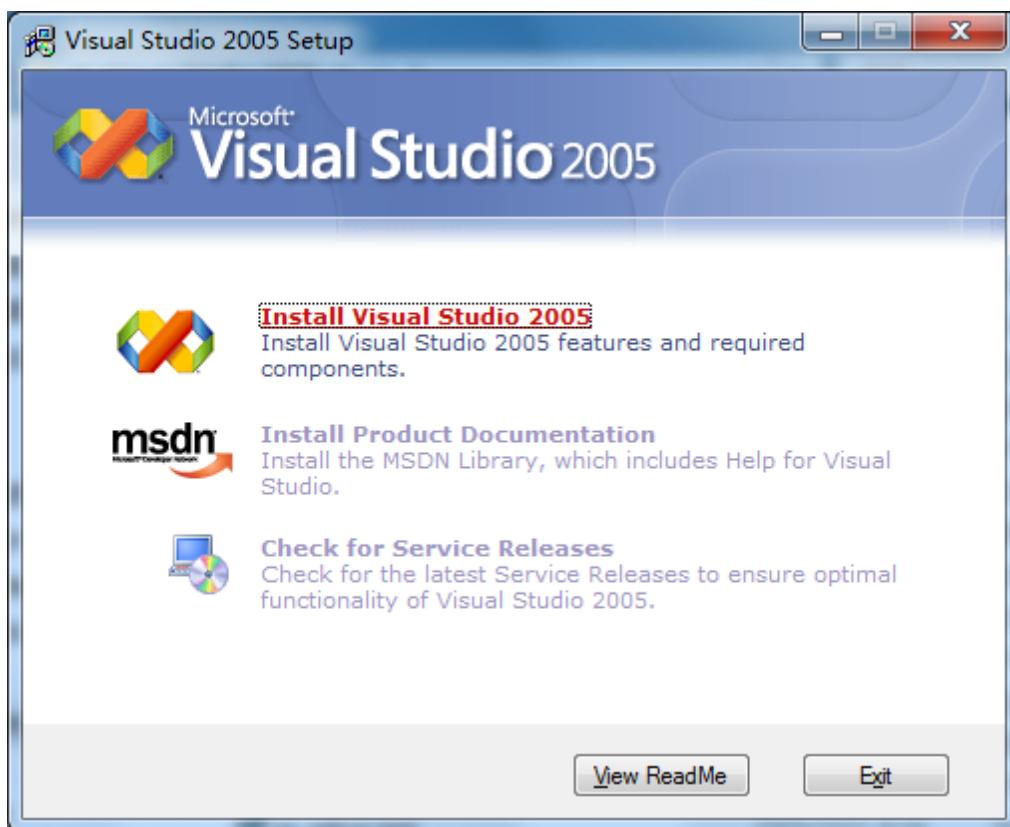
Here are the steps:

5.2.1 Install Visual Studio 2005 and Patches

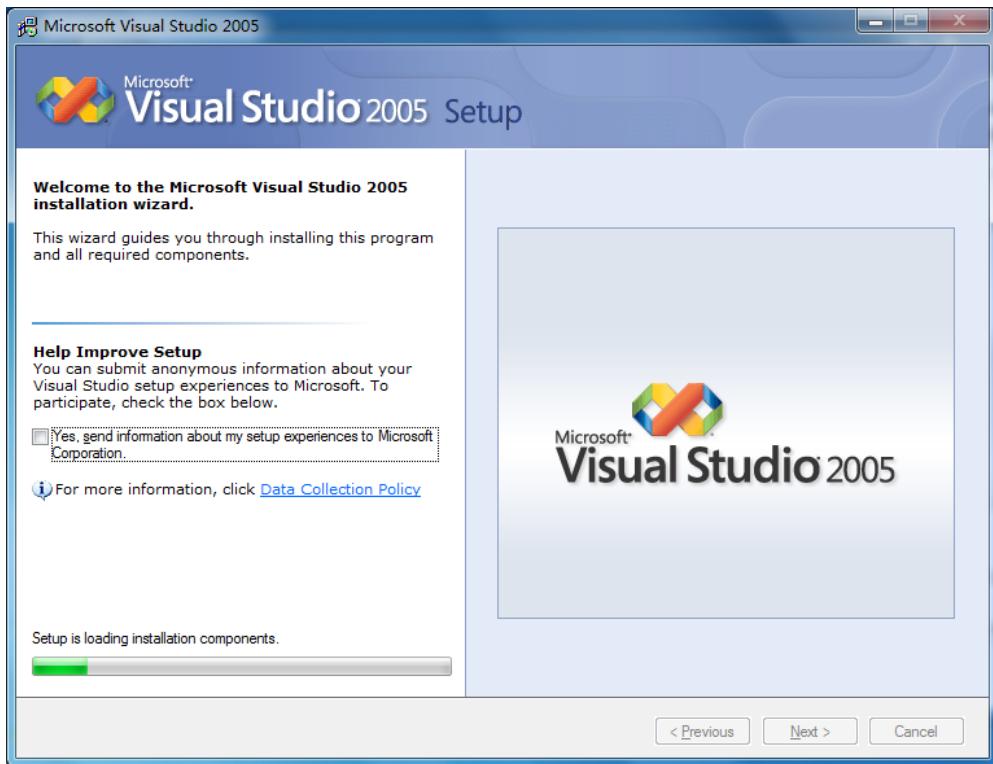
Step1: enter the Visual Studio 2005 directory and double click on “setup.exe”



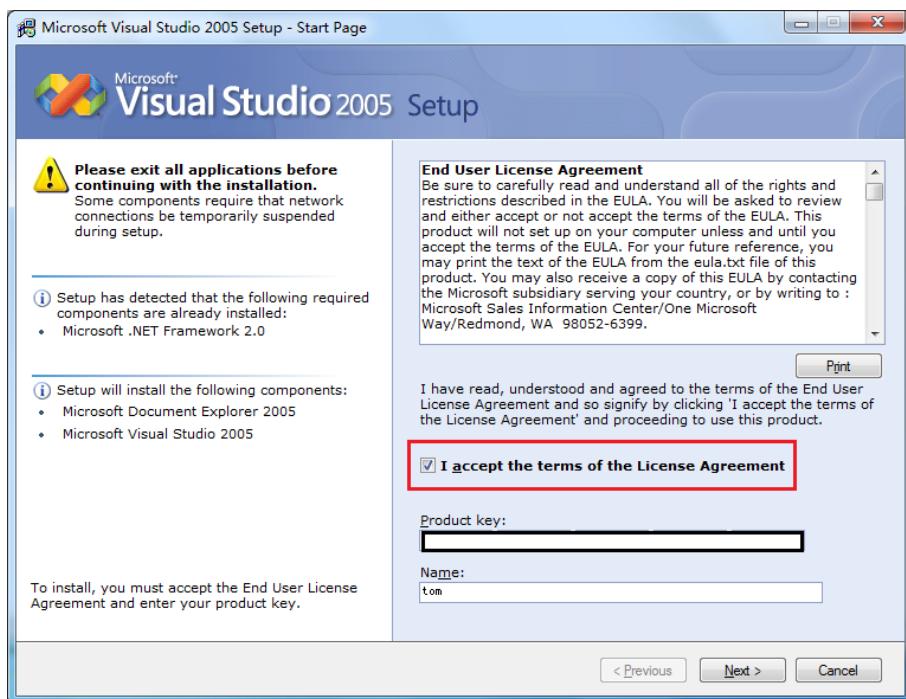
Step2: click on “Install Visual Studio 2005” to continue



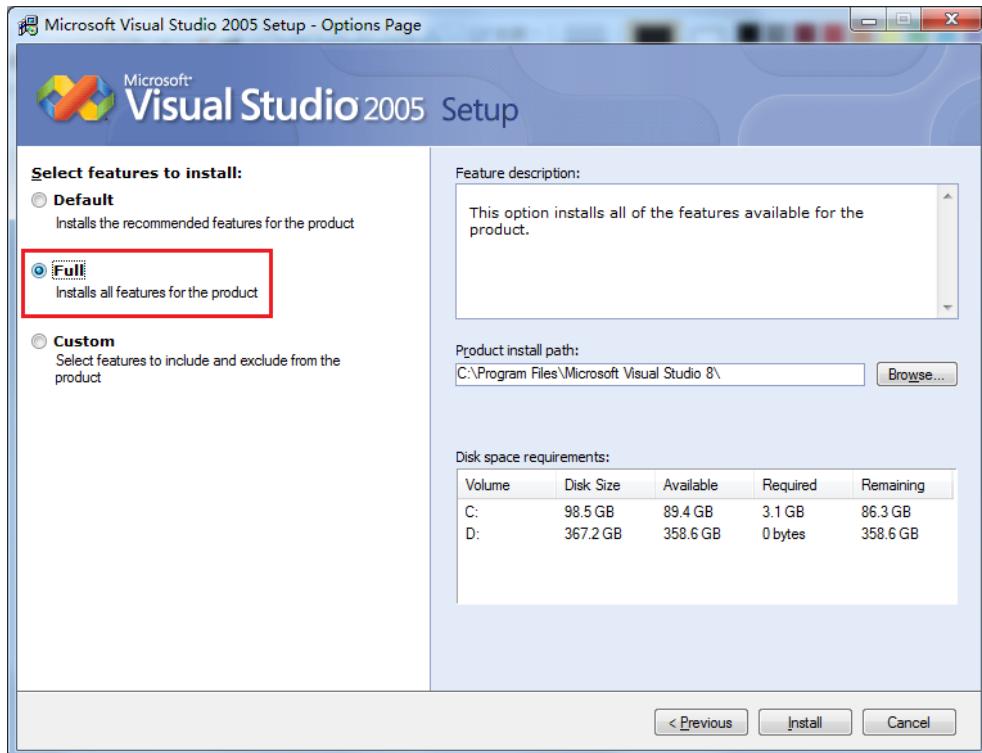
Step3: click on “Next” to continue



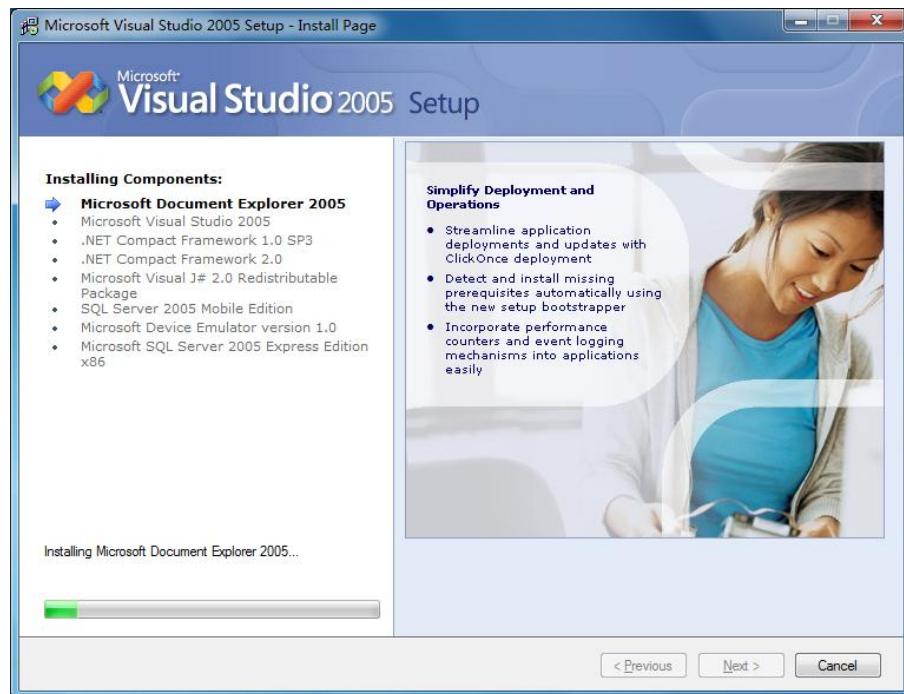
Step4: check the redly marked box and enter your serial number, click on “Next” to continue



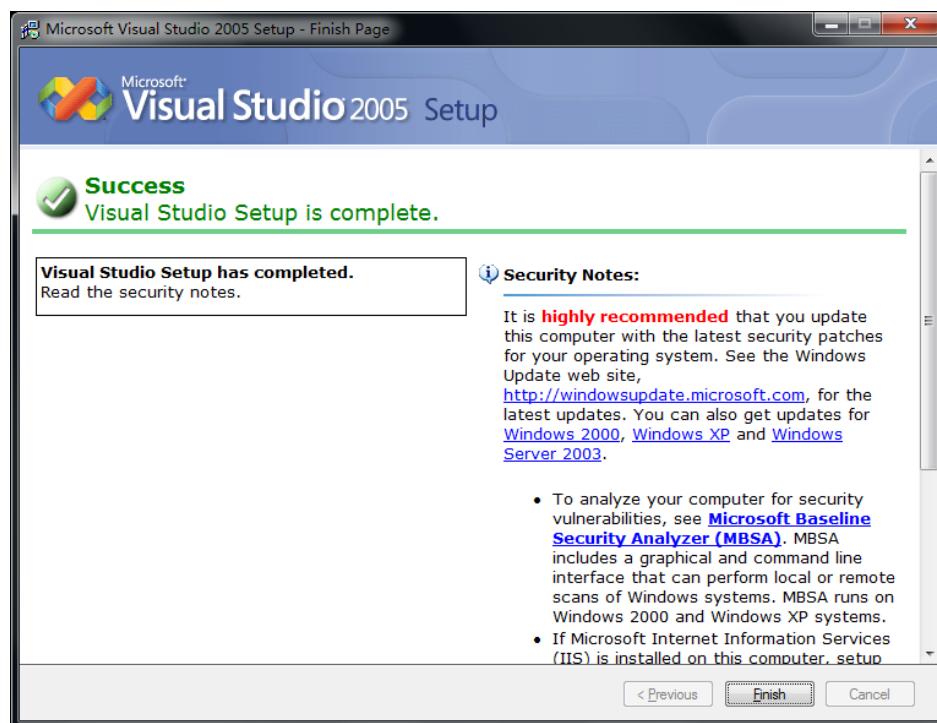
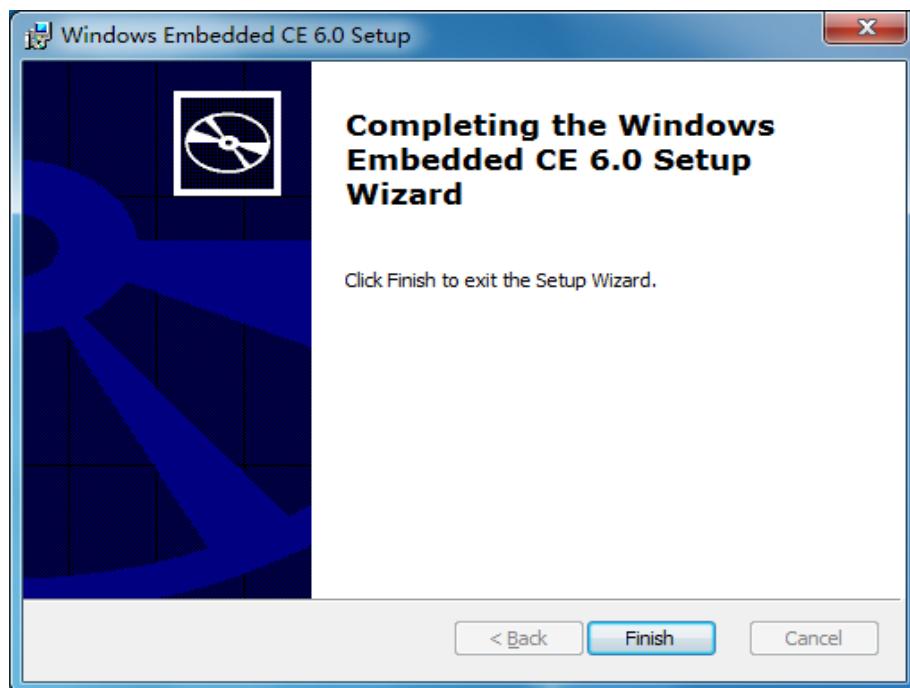
Step5: select your installation features, please check “Full” and click on “Next” to continue



Step6: the following screen kicks off the installation of Visual Studio 2005



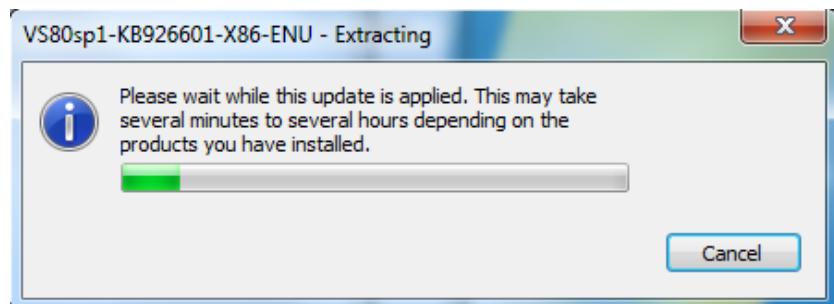
Step7: after Visual Studio 2005 installation is done please click on “Finish” on the dialog shown below



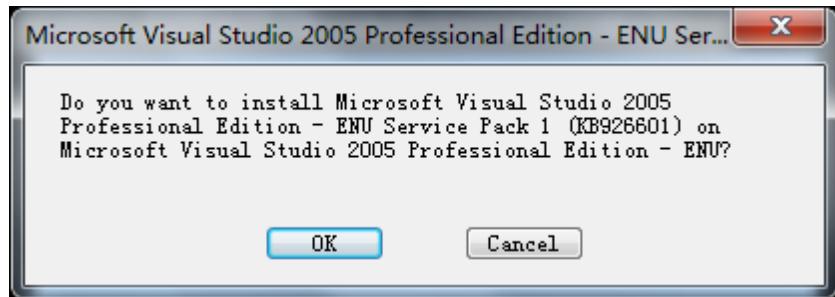
Click on “Exit” on the dialog shown below



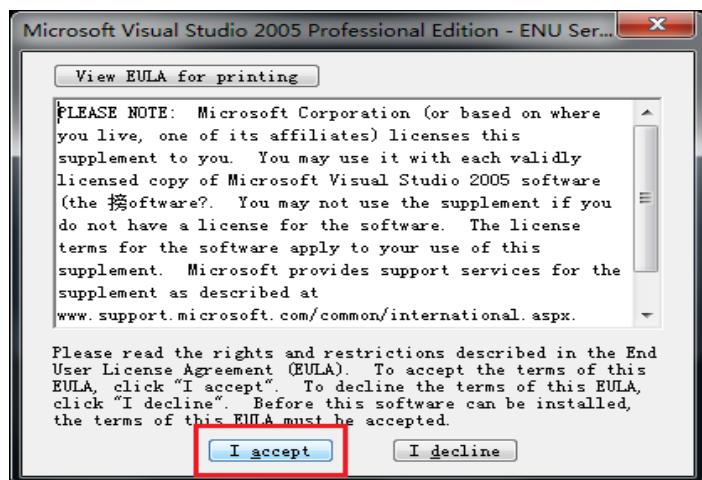
Step8: now we will begin to install “Visual Studio 2005 Service Pack 1”. Double click on “VS80sp1-KB926601-X86-ENU.exe” to begin installation



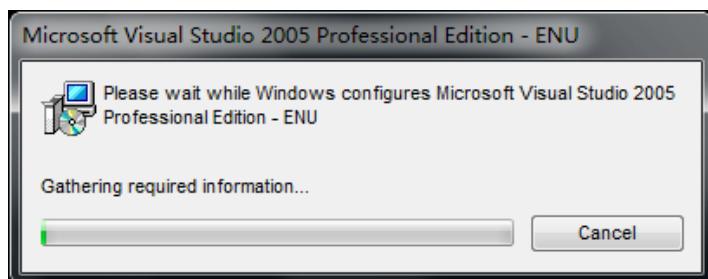
Step9: click on “OK” on the following dialog



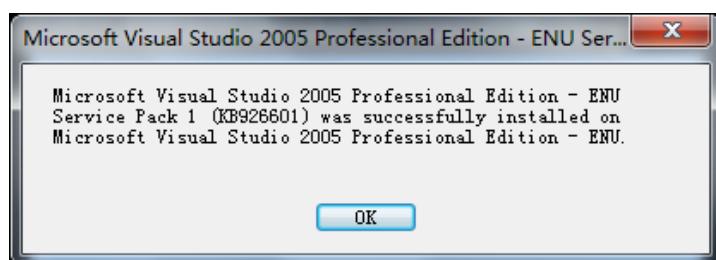
Step10: click on “I accept” on the license dialog



Step11: the following dialog kicks off the installation. The whole process may take a while.

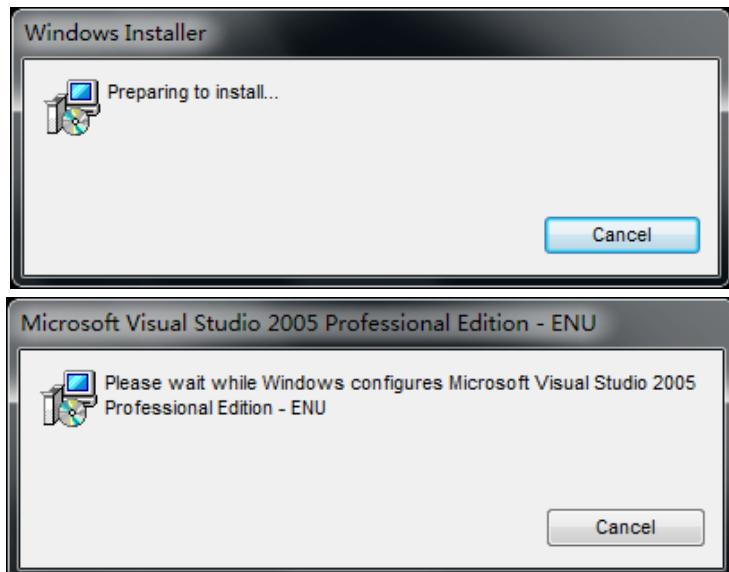


Step12: after installation is done click on “OK” on the following dialog

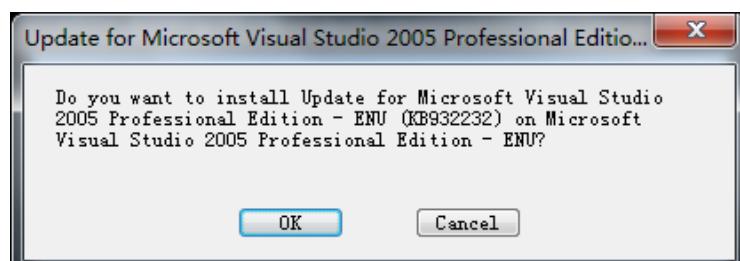


Step13: now we will begin to install the second patch “Visual Studio 2005 Service Pack 1

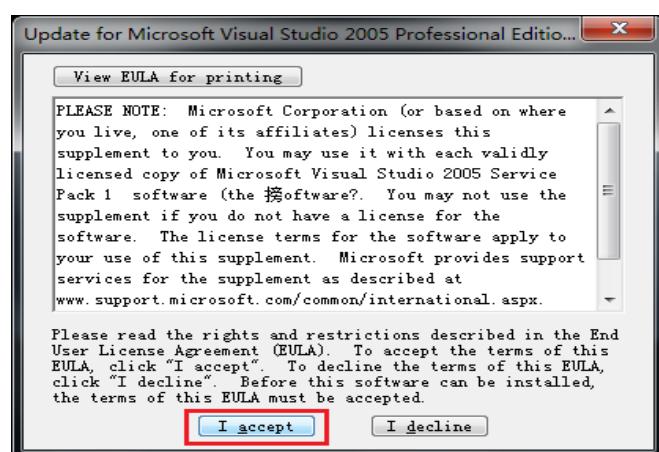
Update for Windows Vista”. Double click on “VS80sp1-KB932232-X86-ENU.exe”



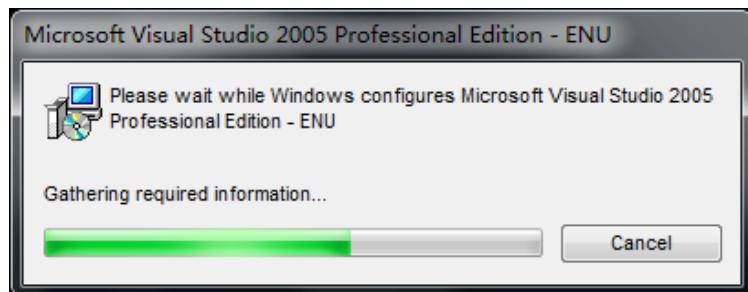
Step14: click on “OK” to continue



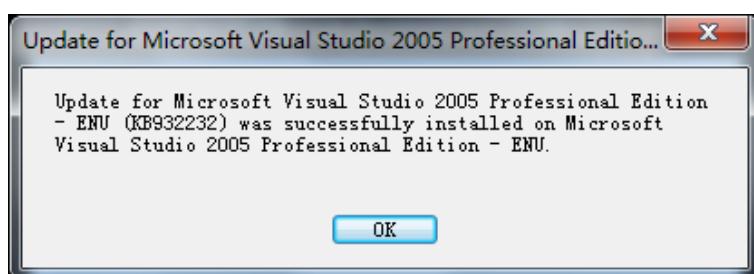
Step15: on the following license dialog click on “I accept”



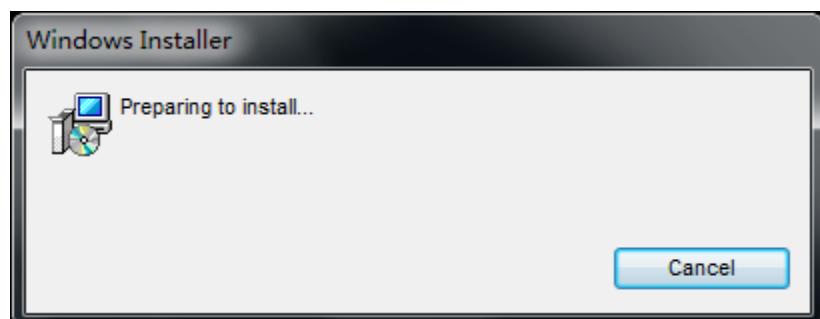
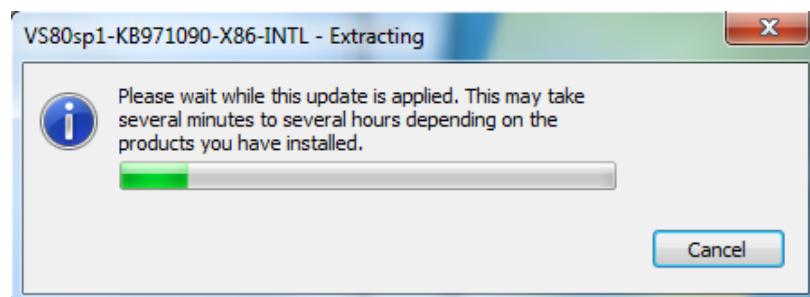
Step16: the following screen kicks off the installation. This may take a while

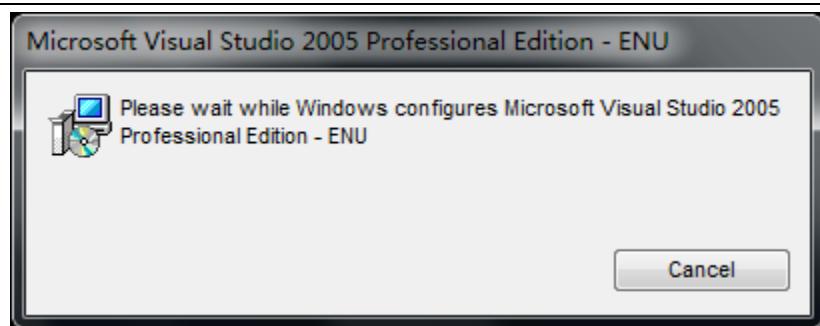


Step17: after the installation is done click on “OK” to finish



Step18: now we will begin to install the third patch “Visual Studio 2005 Service Pack 1 ATL Security Update”. Double click on “VS80sp1-KB971090-X86-INTL.exe”

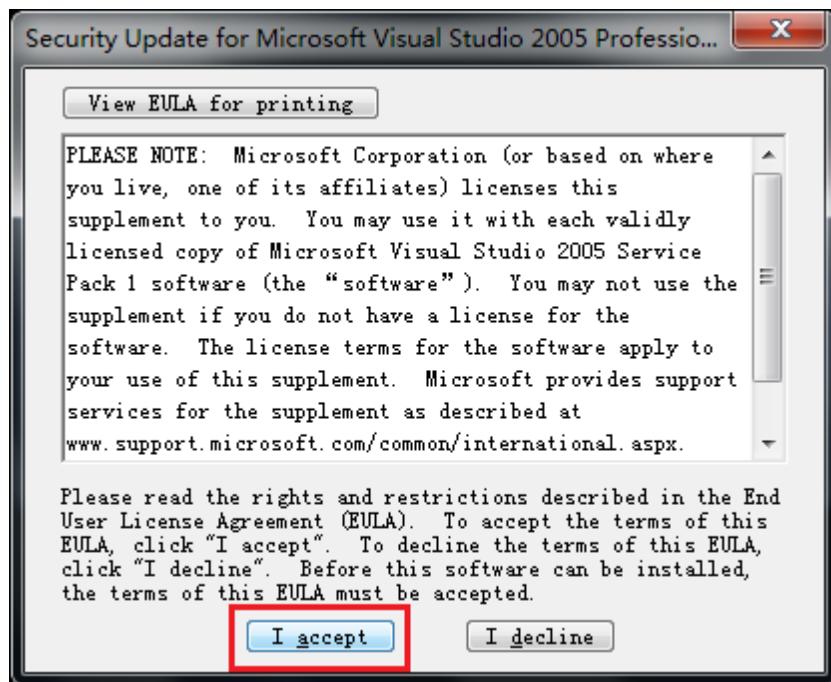




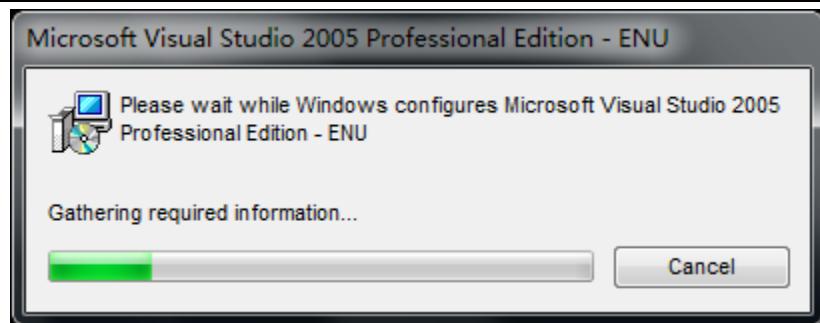
Step19: click on “OK” to continue



Step20: on the license dialog click on “I accept”



Step21: the following screen kicks off the installation. This may take a while



Step22: after the installation is done click on “OK” to finish



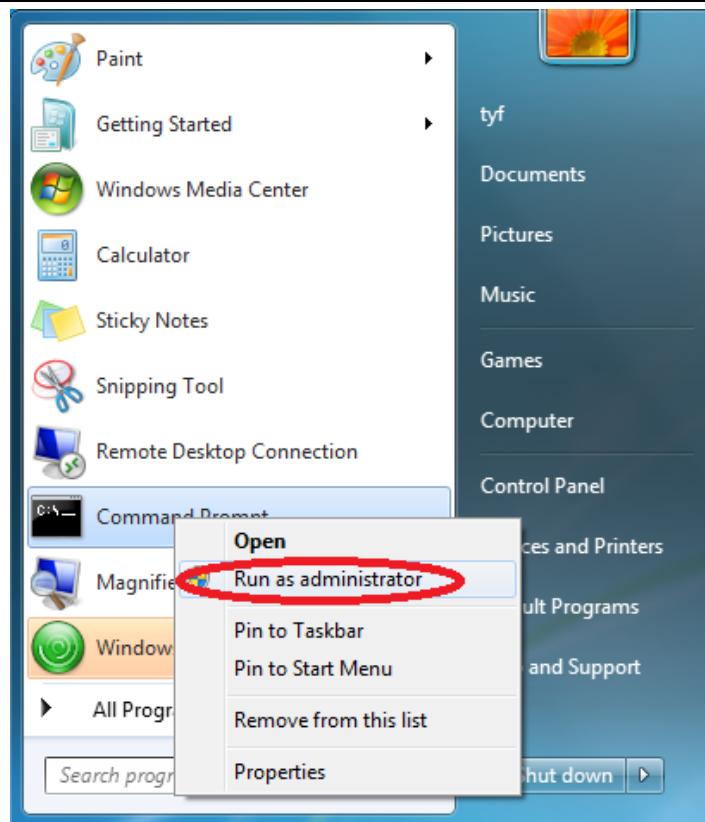
We have completed our installation of Visual Studio 2005 and its patches on Windows7

5.2.2 Install WindowsCE 6.0 and Patches

This section will guide you through the steps on how to install Windows CE 6.0 Platform Builder.

Note: to install Windows CE 6.0 and its patches on Windows 7 you need to do it as administrator. You cannot just double click on the setup.exe to install. Please strictly follow the steps below

Step1: go to “Start”->“Programs”->“Accessories”, locate the command line utility, right click on it and select “run as administrator”



Step2: enter the installation directory, type “Windows Embedded CE 6.0.msi” and press “return”

```

Administrator: Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd "Windows CE 6.0"
The system cannot find the path specified.

C:\Windows\system32>cd D;
The system cannot find the path specified.

C:\Windows\system32>cd d:
D:\

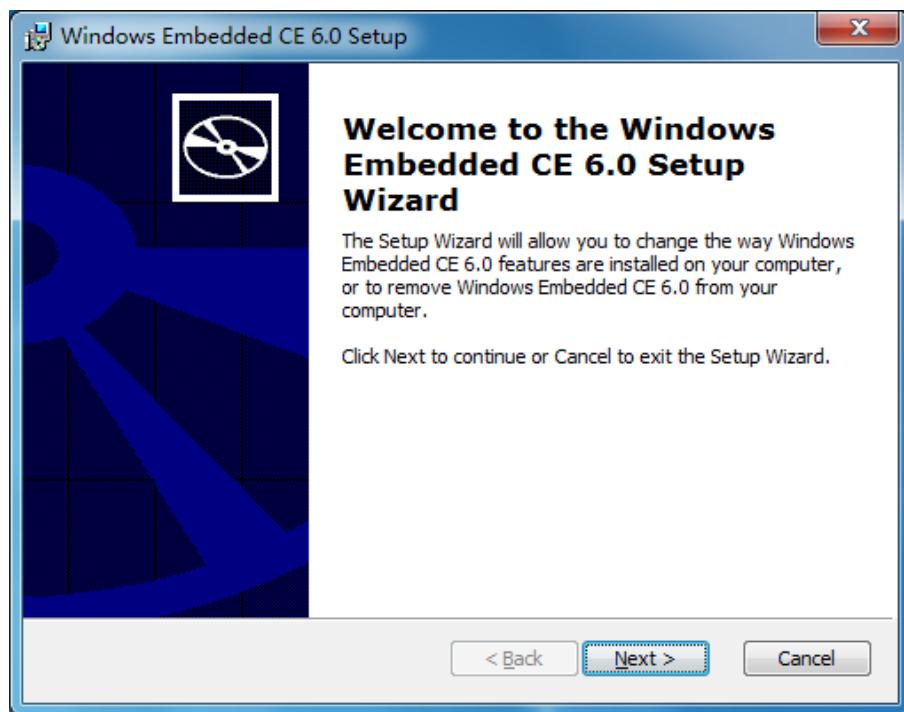
C:\Windows\system32>d:

D:\>cd "WindowsCE6 Installation"

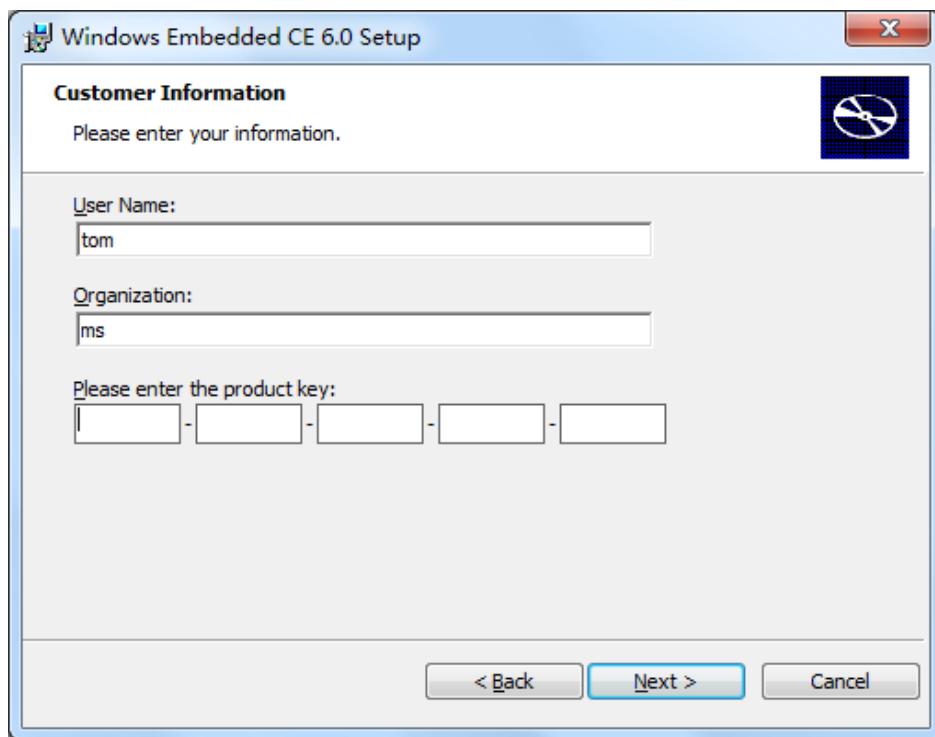
D:\WindowsCE6 Installation>"Windows Embedded CE 6.0.msi"

```

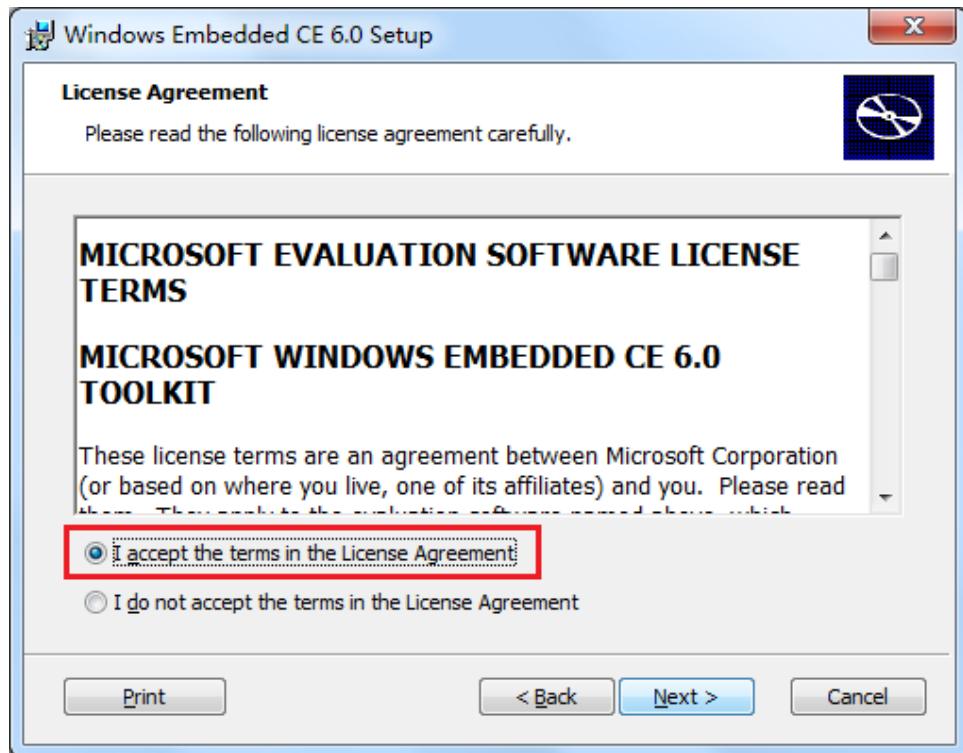
Step3: click on “Next” to continue



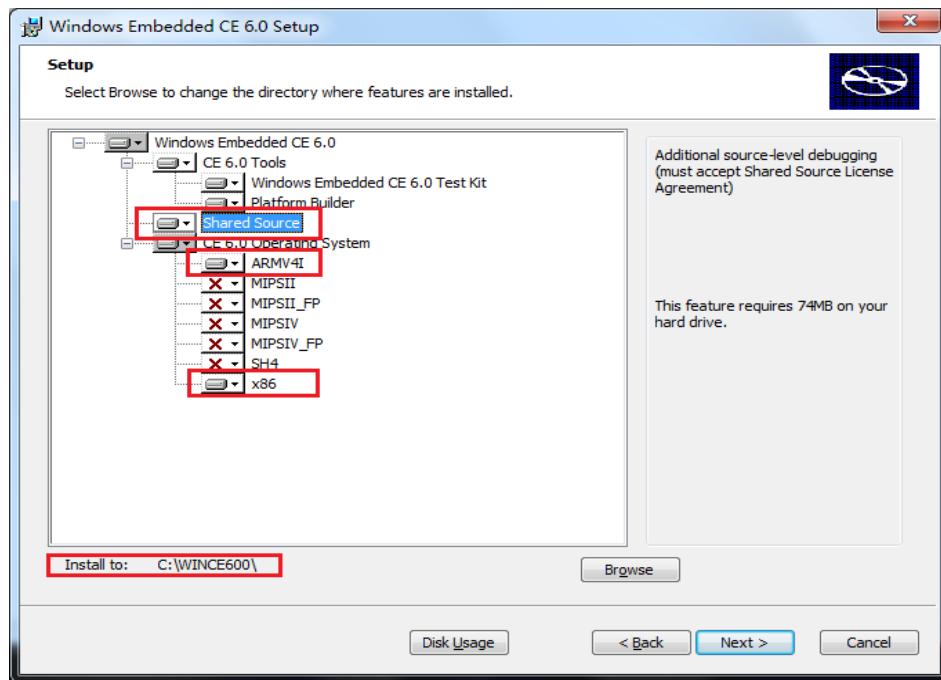
Step4: type the product key and click on “Next”



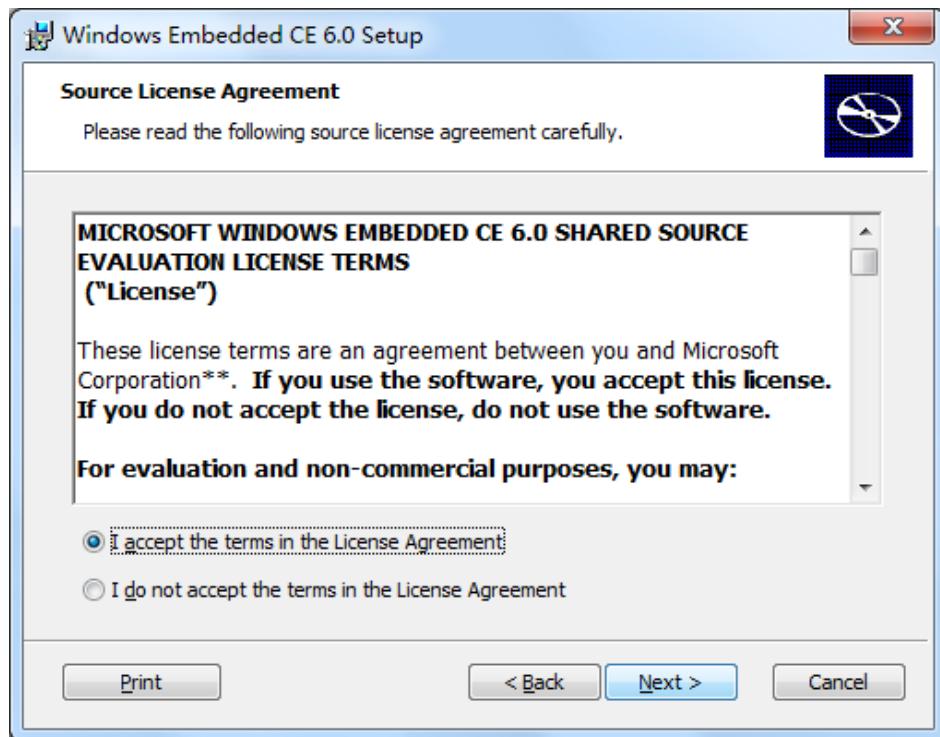
Step5: check “I accept” on the license dialog and click on “Next” to continue



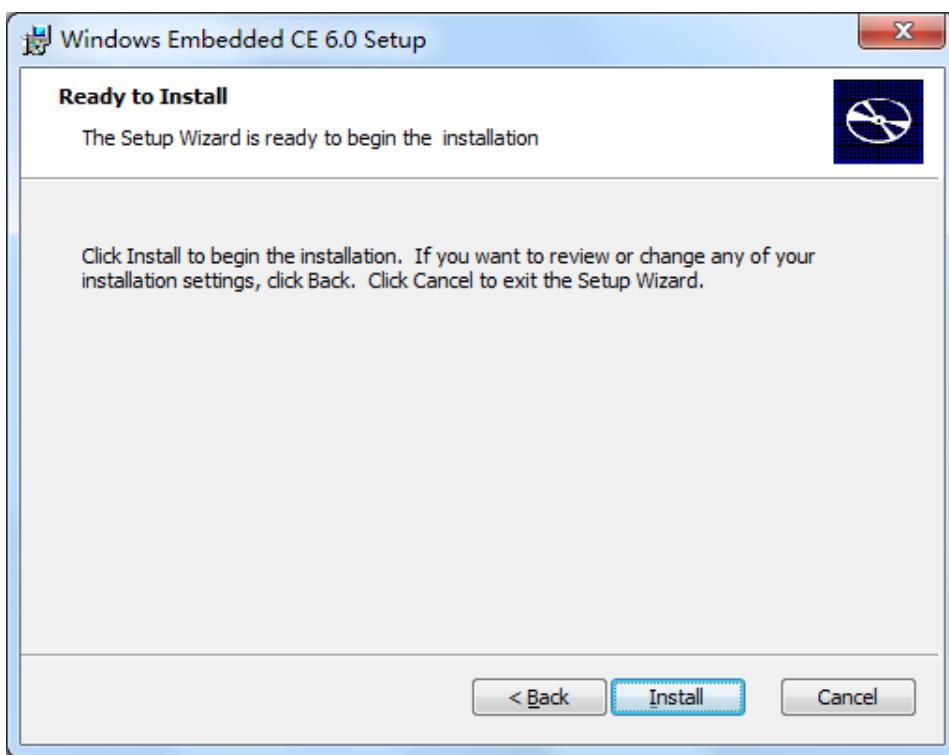
Step6: follow the options marked redly as below and click on “Next” to continue



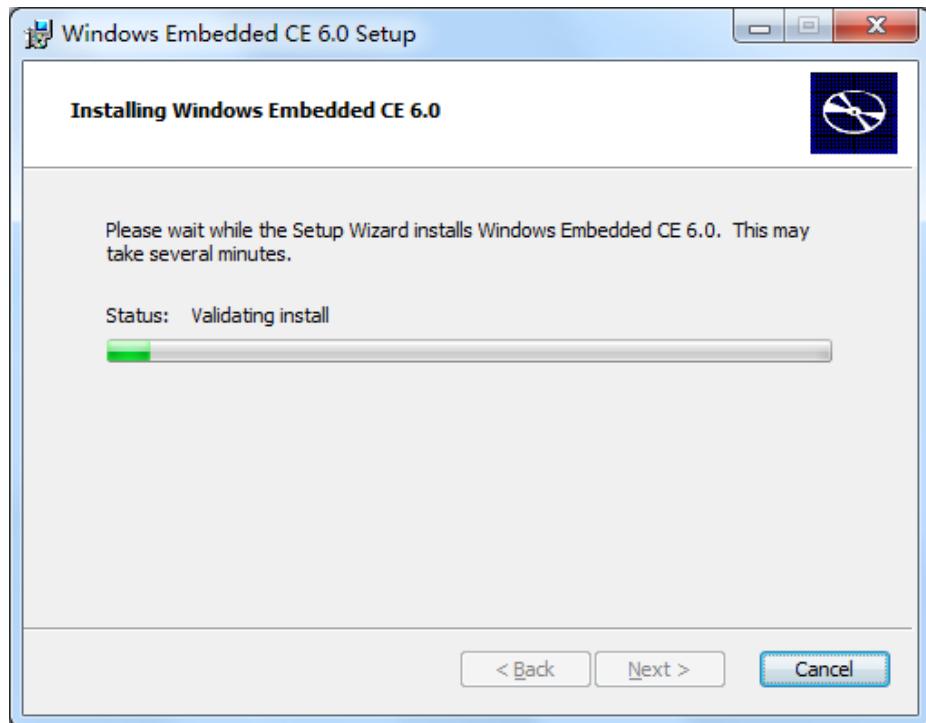
Step7: follow the options mared redly as below and click on “Next” to continue



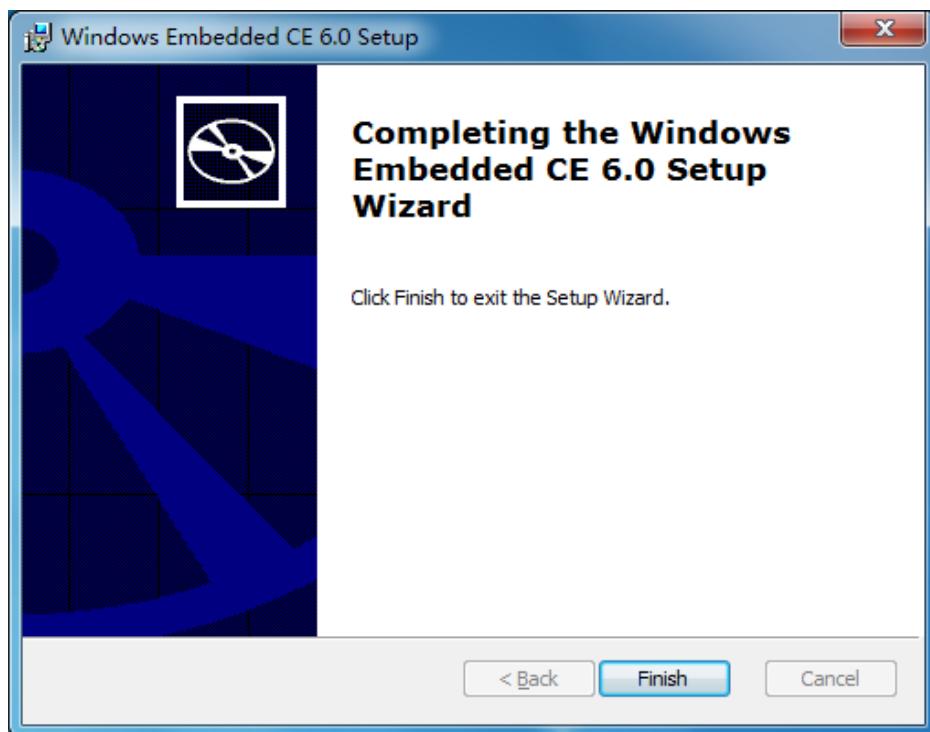
Step8: click on “Install” to continue



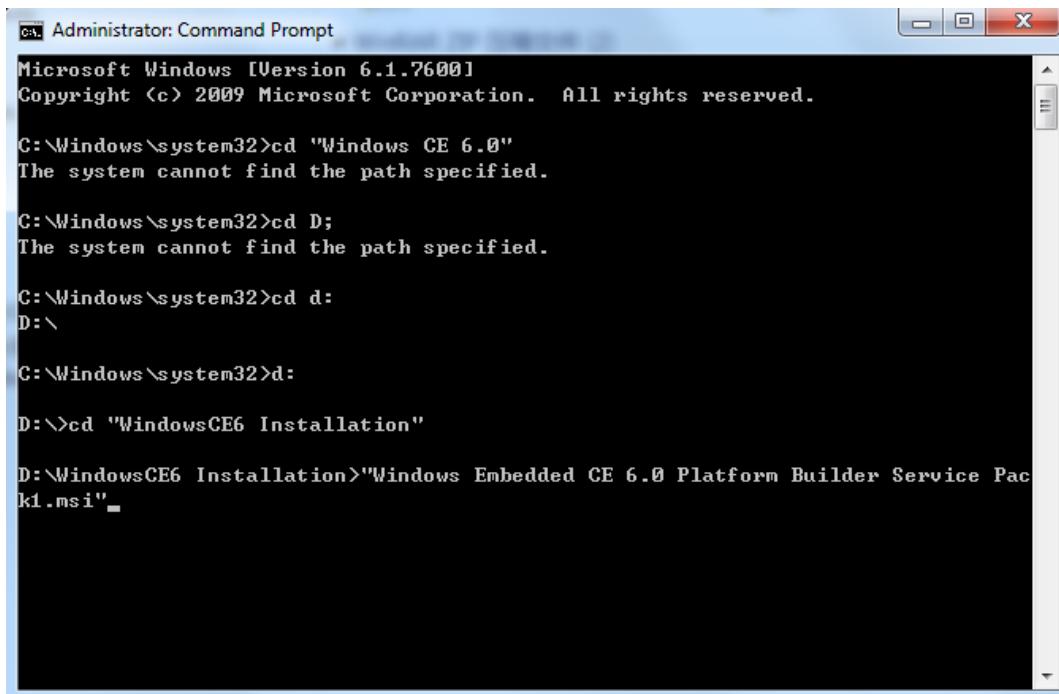
Step9: the installation is kicked off and it may take a while



Step10: after the installation is done click on “Finish” to complete



Step11: now we will begin to install a patch “Windows Embedded CE 6.0 Platform Builder Service Pack 1.msi”. Open the command line utility as administrator, enter the directory, type “Windows Embedded CE 6.0 Platform Builder Service Pack 1.msi” and press “return”



```
C:\ Administrator: Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd "Windows CE 6.0"
The system cannot find the path specified.

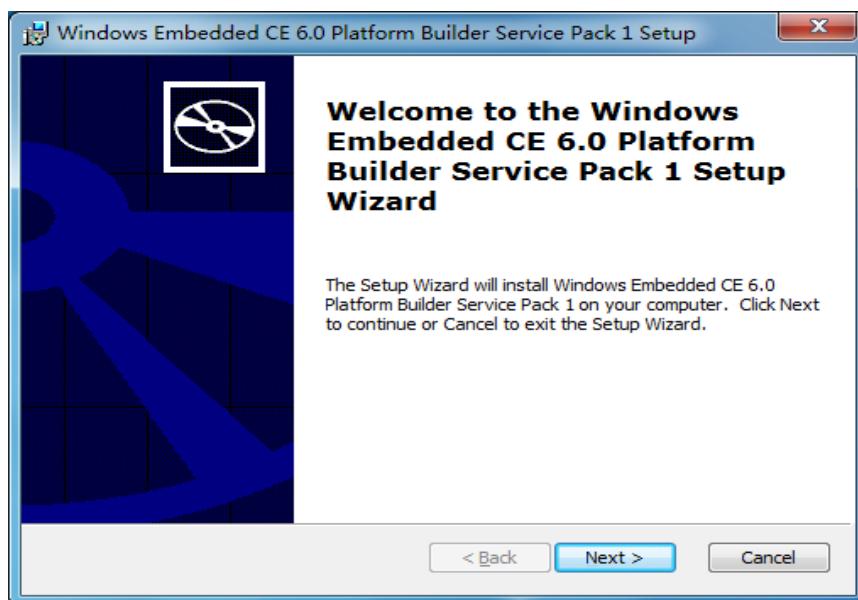
C:\Windows\system32>cd D:
The system cannot find the path specified.

C:\Windows\system32>cd d:
D:\

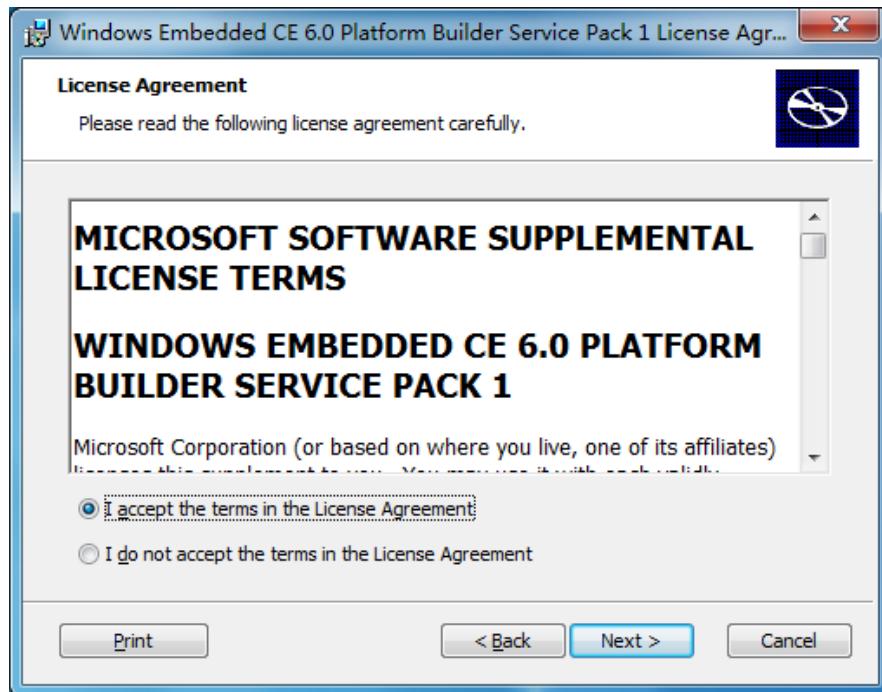
C:\Windows\system32>d:
D:\>cd "WindowsCE6 Installation"

D:\WindowsCE6 Installation>"Windows Embedded CE 6.0 Platform Builder Service Pack 1.msi"
```

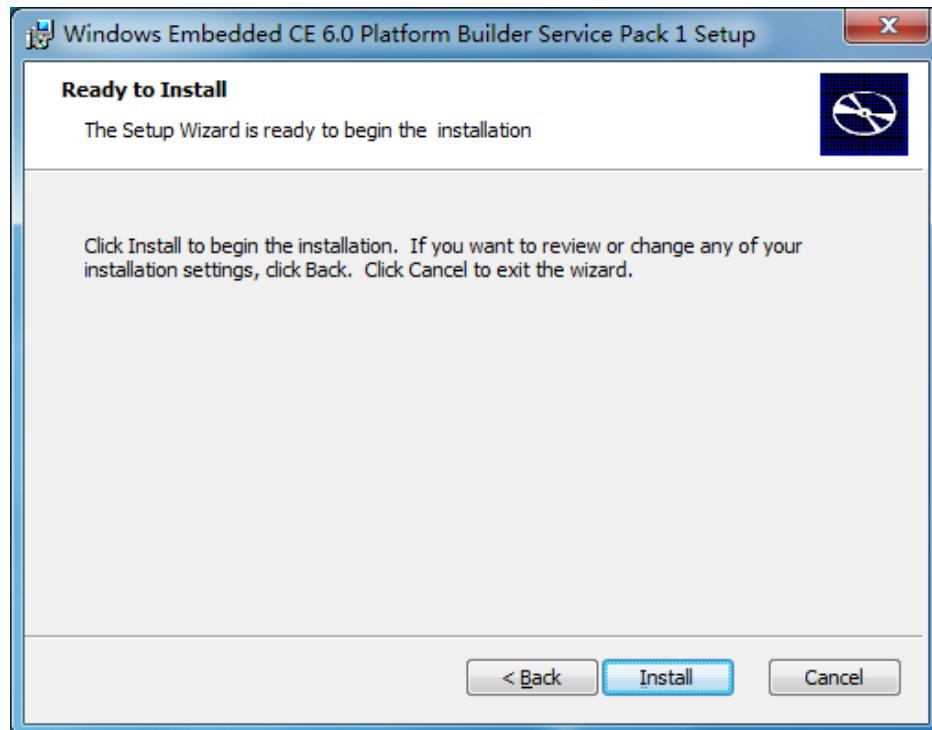
Step12: click on “Next” to continue



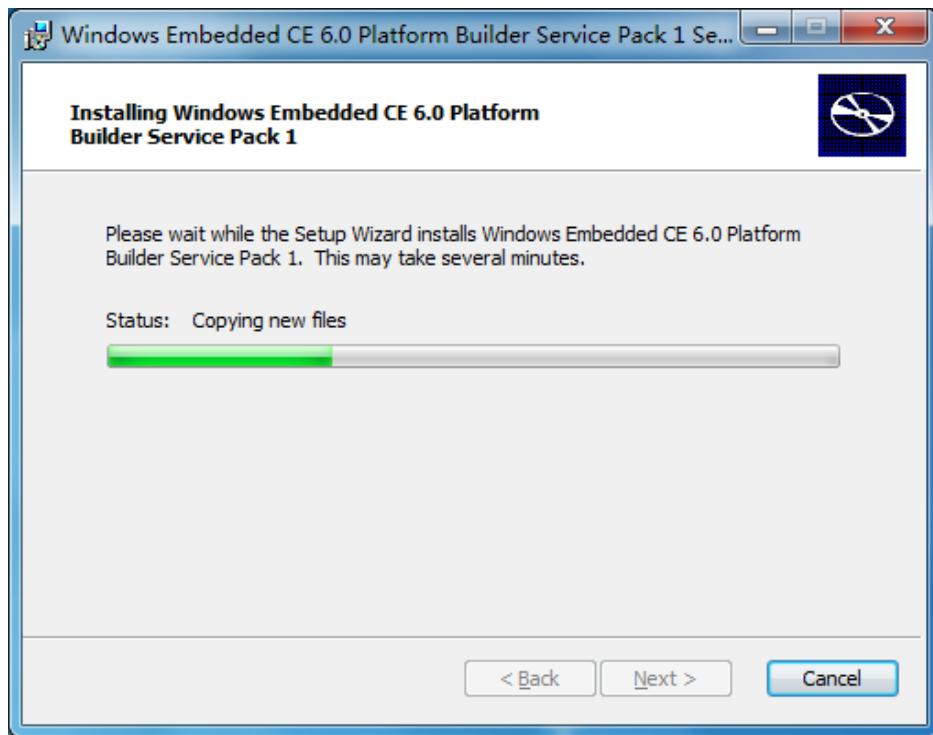
Step13: check “I accept” and click on “Next” to continue



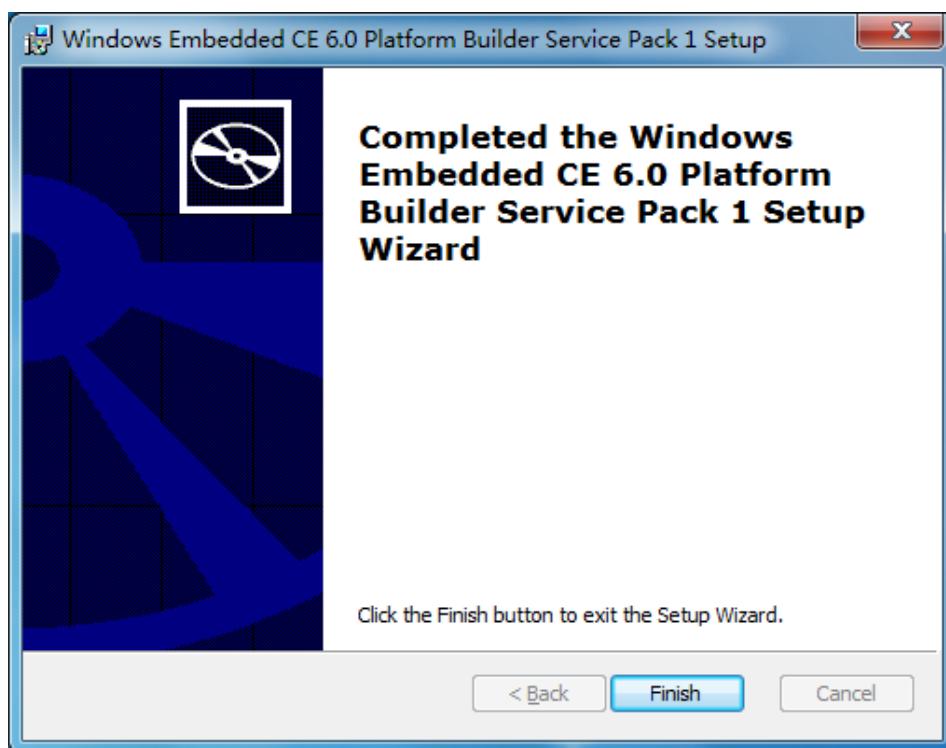
Step14: click on “Next” to continue



Step15: the installation is kicked off and it may take a while



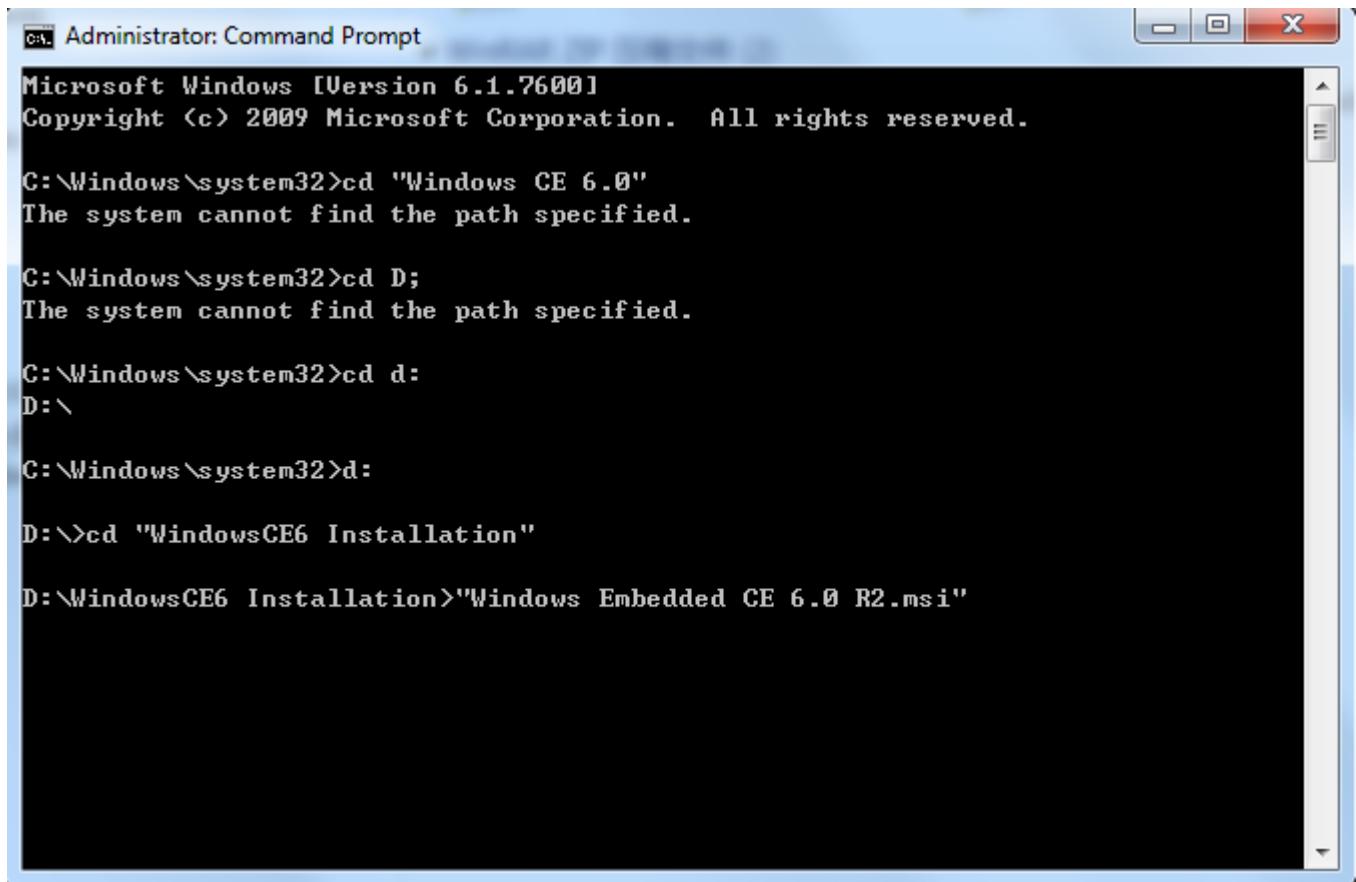
Step16: after the installation is done, click on “Finish” to complete



Step17: now we will begin to install the second patch “Windows Embedded CE 6.0 R2.msi”.

Please follow the previous steps to launch the command line utility, type “Windows Embedded CE 6.0 R2.msi” and press “return”

Note: some users may download a “Windows Embedded CE 6.0 R2.msi” which is about 50MB. But this is not a complete installation file. We suggest our users use our R2 patch which is about 1.01GB



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd "Windows CE 6.0"
The system cannot find the path specified.

C:\Windows\system32>cd D;
The system cannot find the path specified.

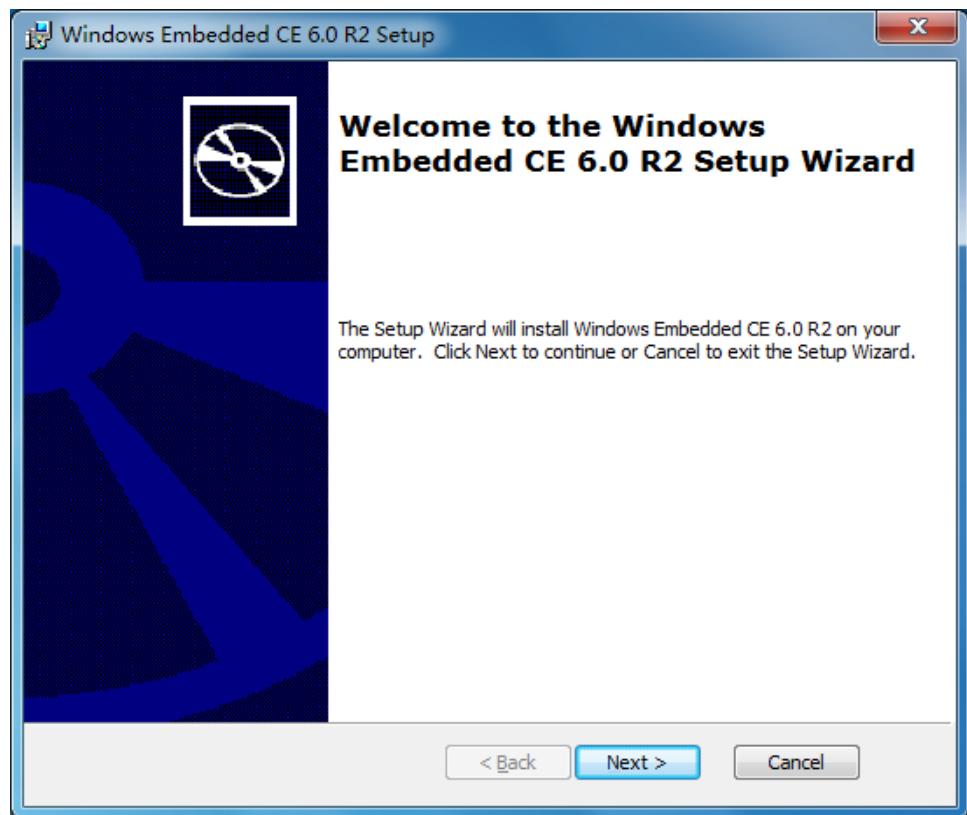
C:\Windows\system32>cd d:
D:\

C:\Windows\system32>d:

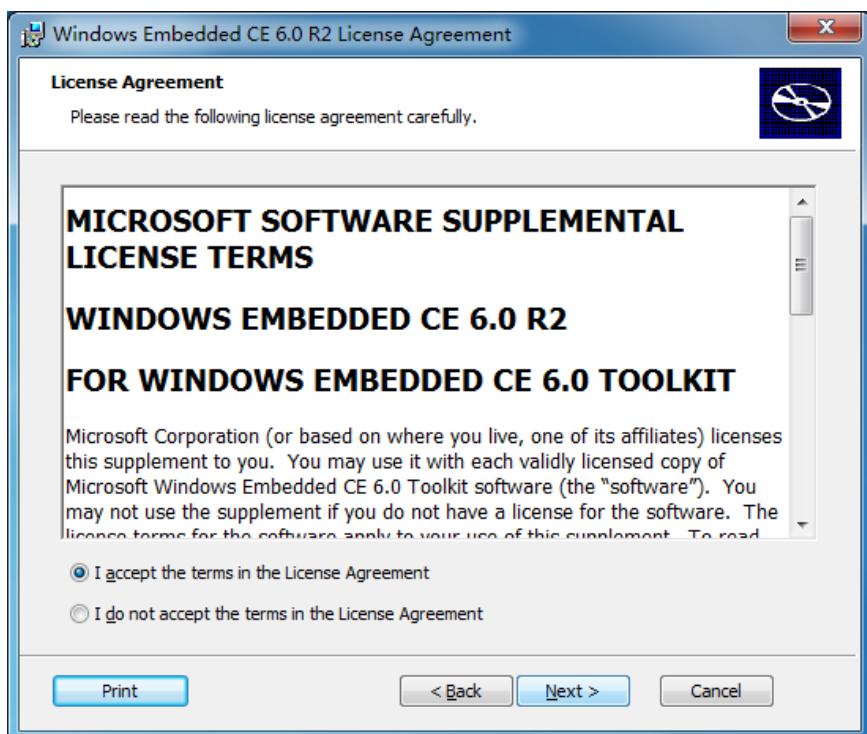
D:\>cd "WindowsCE6 Installation"

D:\WindowsCE6 Installation>"Windows Embedded CE 6.0 R2.msi"
```

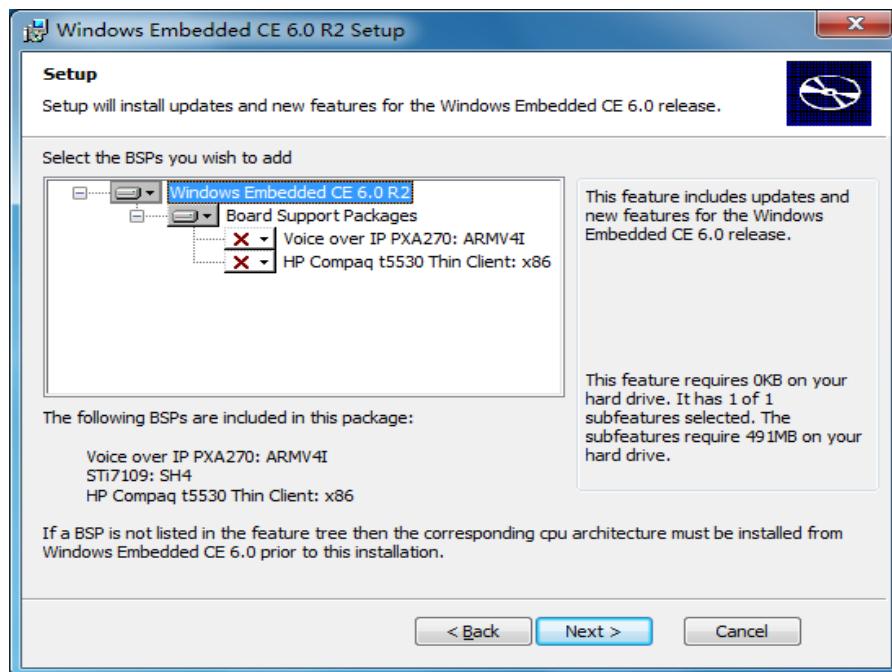
Step18: click on “Next” to continue



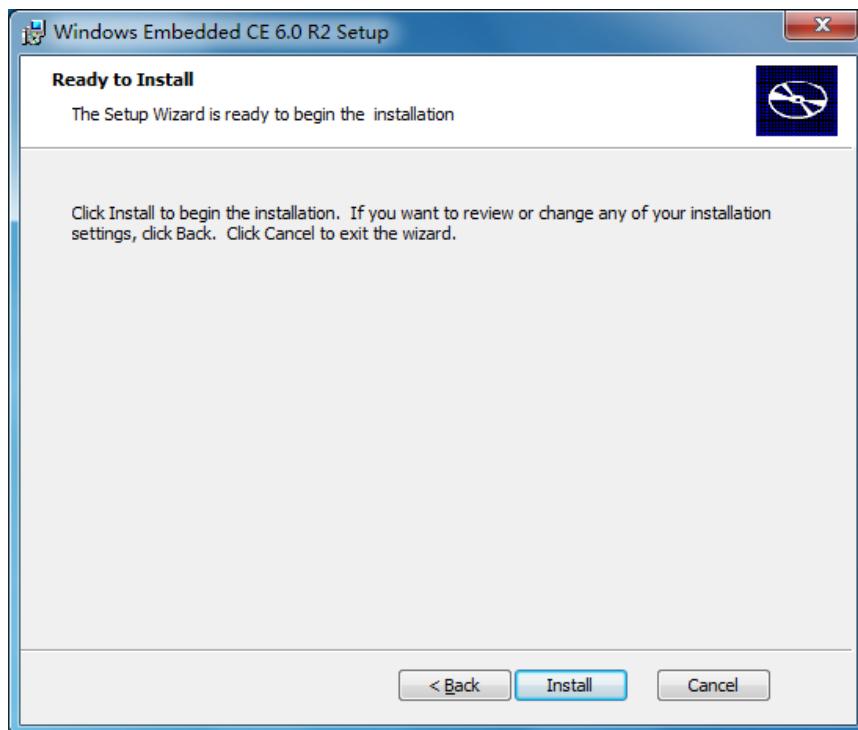
Step19: check “I accept” and click on “Next” to continue



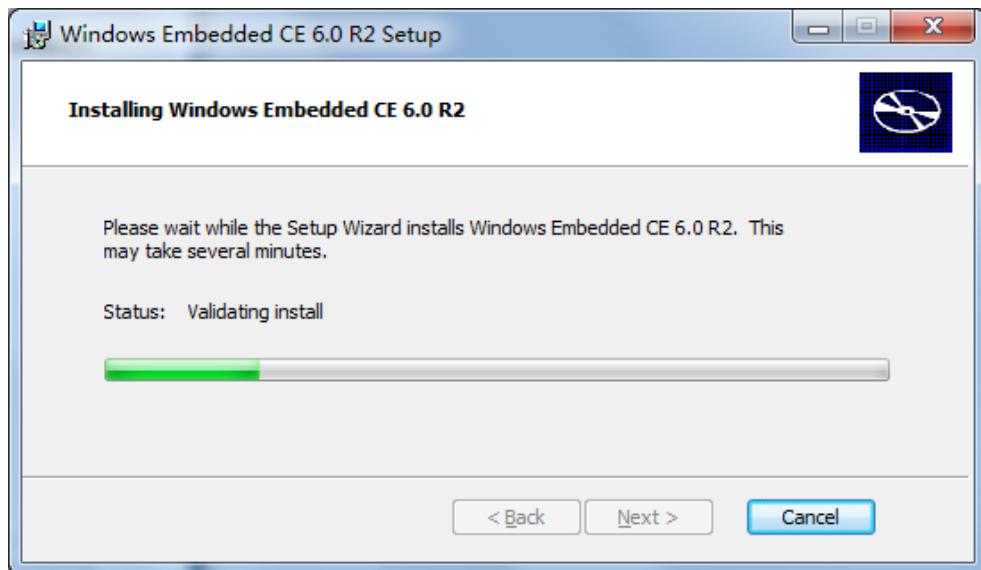
Step20: click on “Next” to continue



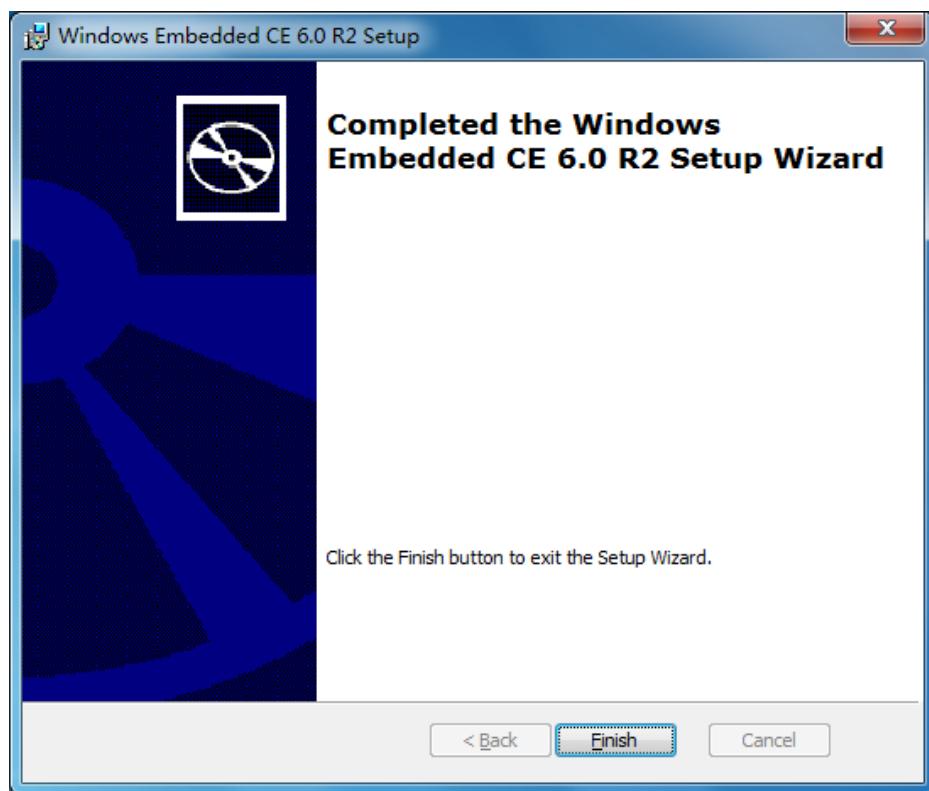
Step21: click on “Next” to continue



Step22: the installation process is kicked off and this may take a while



Step23: after the installation is done, click on “Finish” to complete

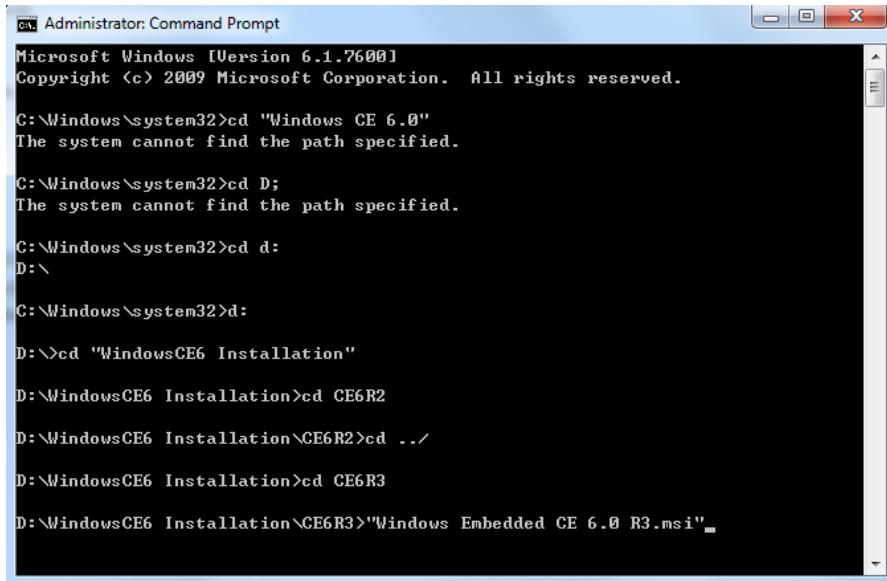


Step24: now we will begin to install R3. Please follow the previous steps to launch the

command line utility, type “Windows Embedded CE 6.0 R2.msi” and press “enter”

Note: some users may download a “Windows Embedded CE 6.0 R3” which is an image file.

In order for users to use it more conveniently we make it a directory which contains 166 files that are about 1.14GB



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

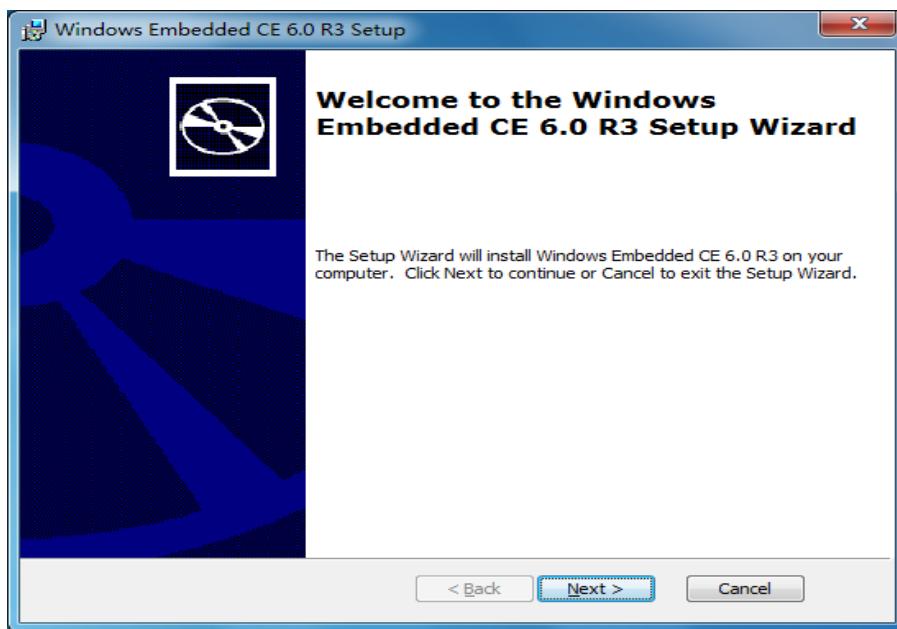
C:\Windows\system32>cd "Windows CE 6.0"
The system cannot find the path specified.

C:\Windows\system32>cd D:
The system cannot find the path specified.

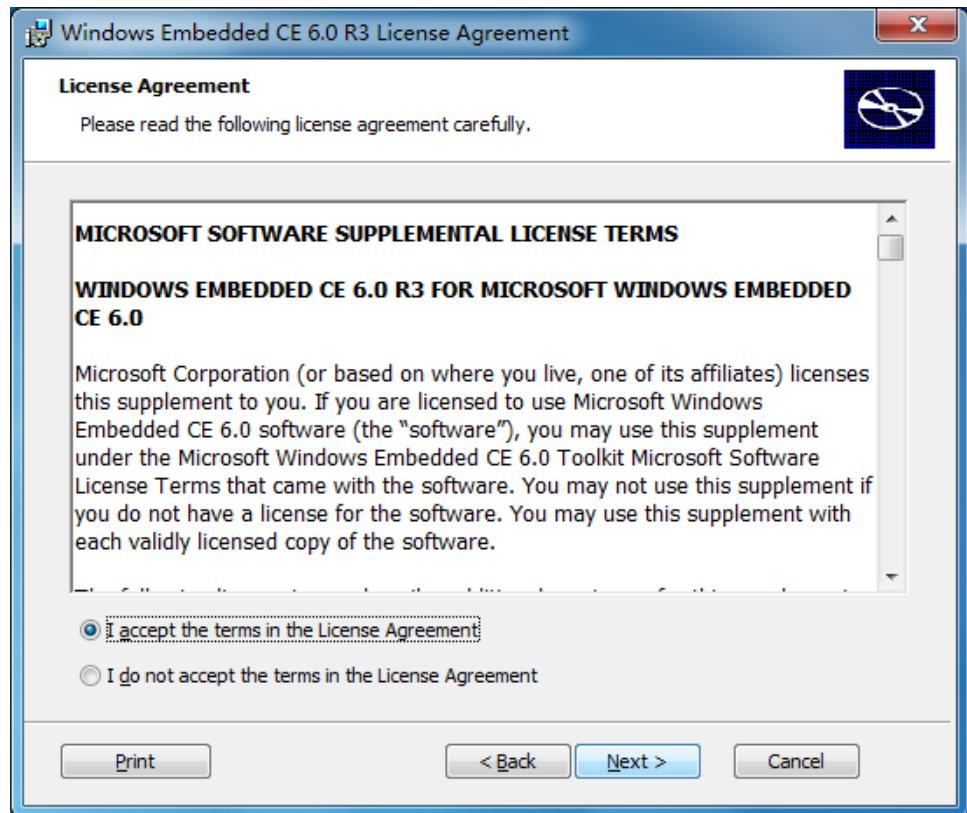
C:\Windows\system32>cd d:
D:<

C:\Windows\system32>d:
D:<>cd "WindowsCE6 Installation"
D:\WindowsCE6 Installation>cd CE6R2
D:\WindowsCE6 Installation\CE6R2>cd ../
D:\WindowsCE6 Installation>cd CE6R3
D:\WindowsCE6 Installation\CE6R3>"Windows Embedded CE 6.0 R3.msi"
```

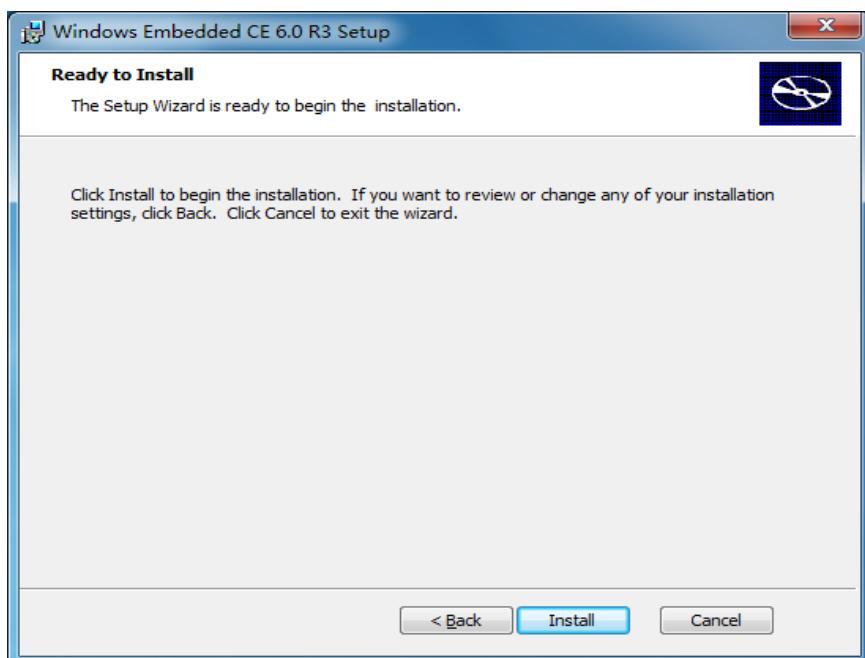
Step25: click on “Next” to continue



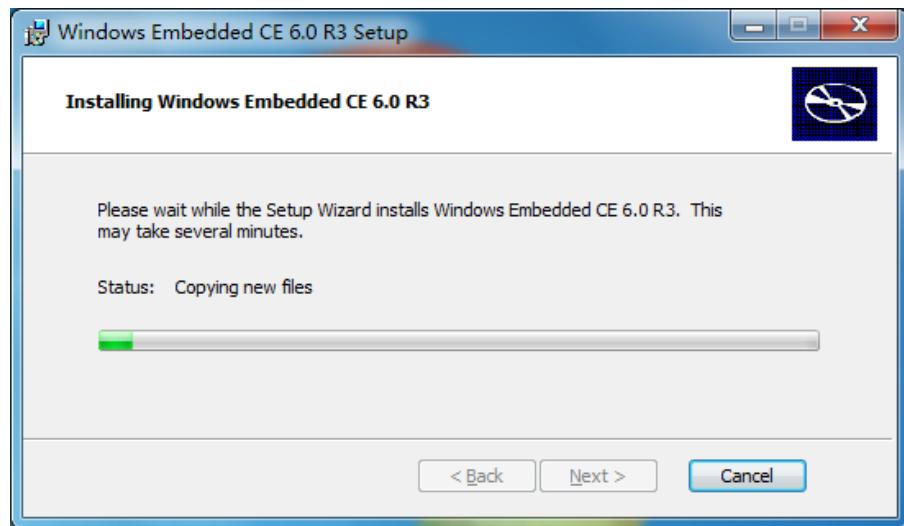
Step26: check “I accept” and click on “Next” to continue



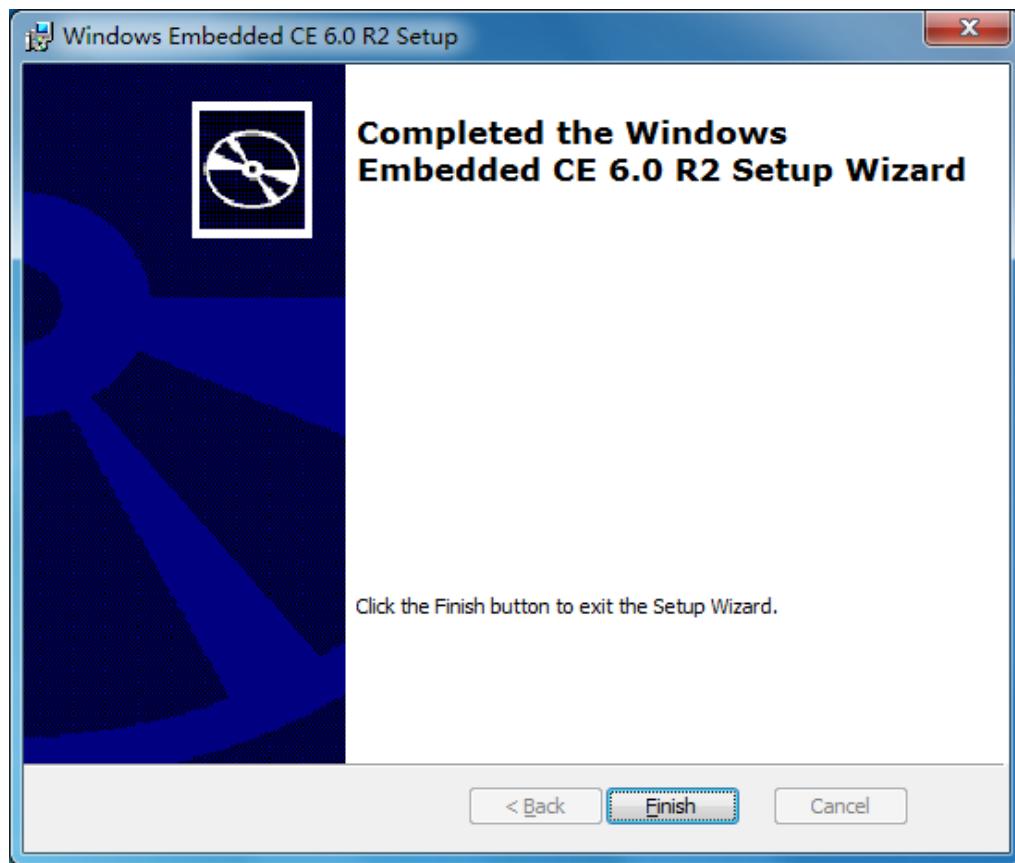
Step27: click on “Next” to continue



Step28: the installation is kicked off and it may take a while



Step29: after the installation is done, click on “Finish” to complete



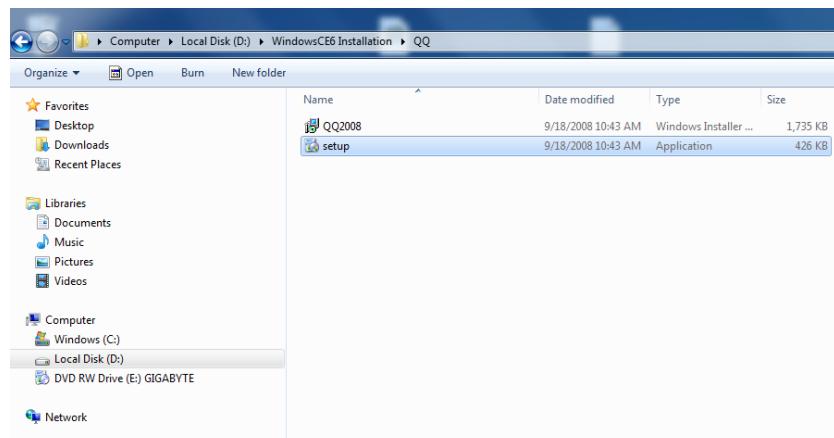
5.2.3 Install Tencent QQ (A Third Party Messenger)

Windows CE 6.0 R3 includes some third party software such as Tencent QQ and File Viewers. We have burned it into our shipped CD and you can also download it from the following website:

<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=bc247d88-ddb6-4d4a-a595-8eee3556fe46#filelist>.

We will intall QQ as an example for you.

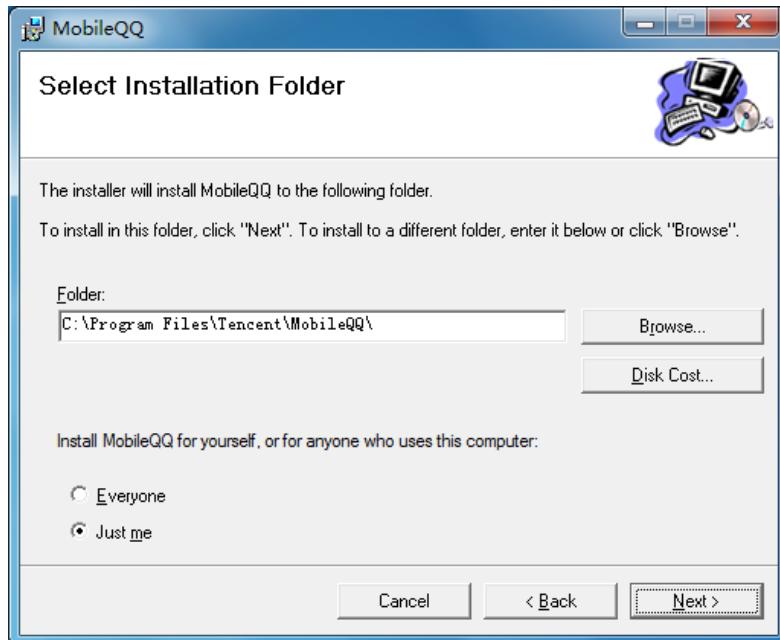
Step1: enter the QQ directory, double click on the “setup.exe” executable.



Step2: click on “Next” to continue

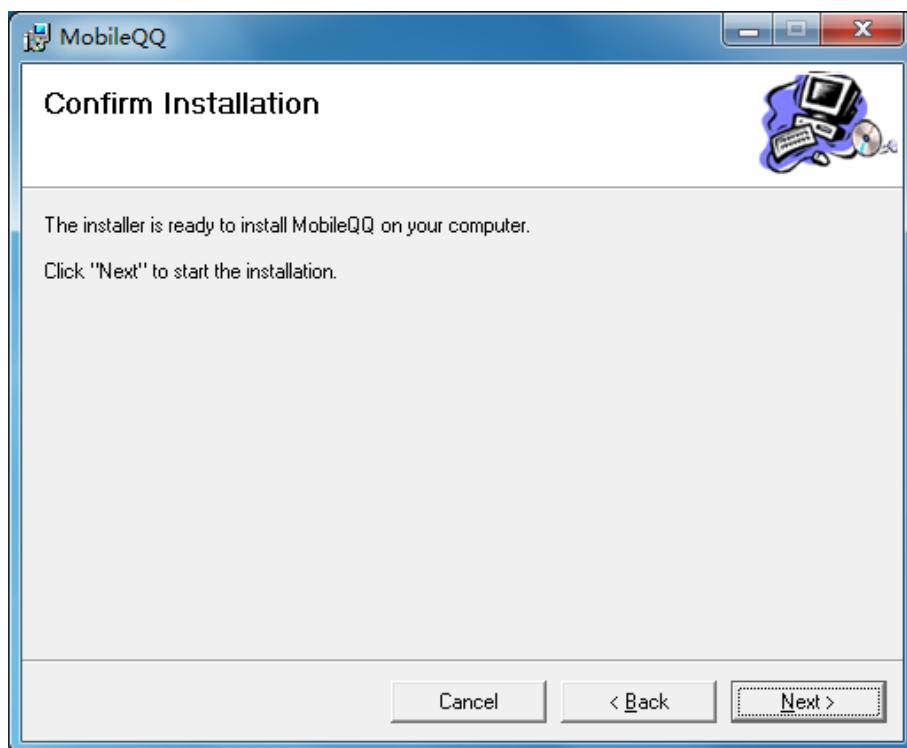


Step3: follow the default options and click on “Next” to continue

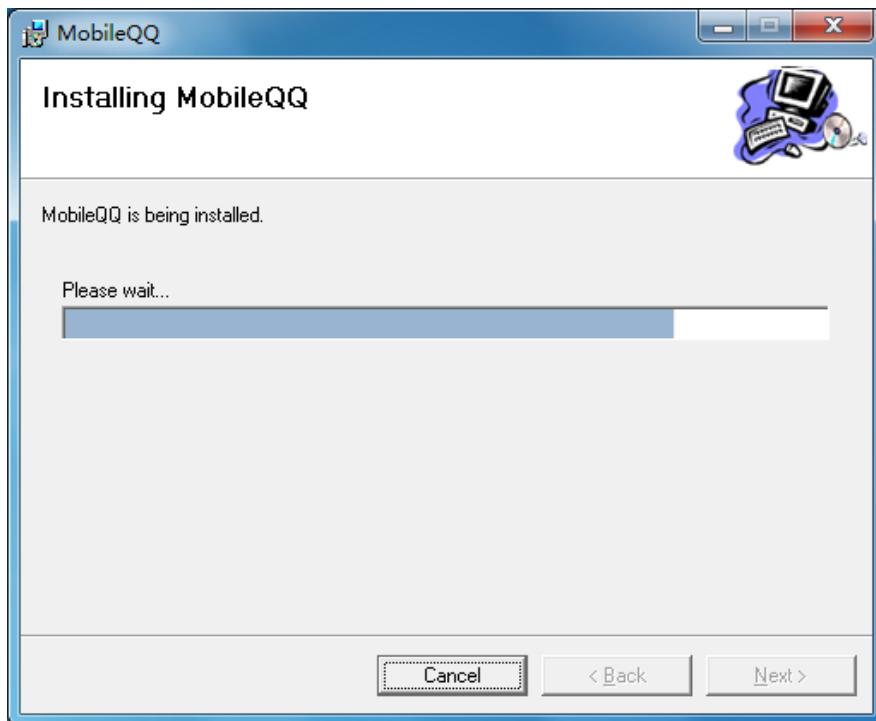


Step4: check “I agree” on the license dialog and click on “Next” to continue

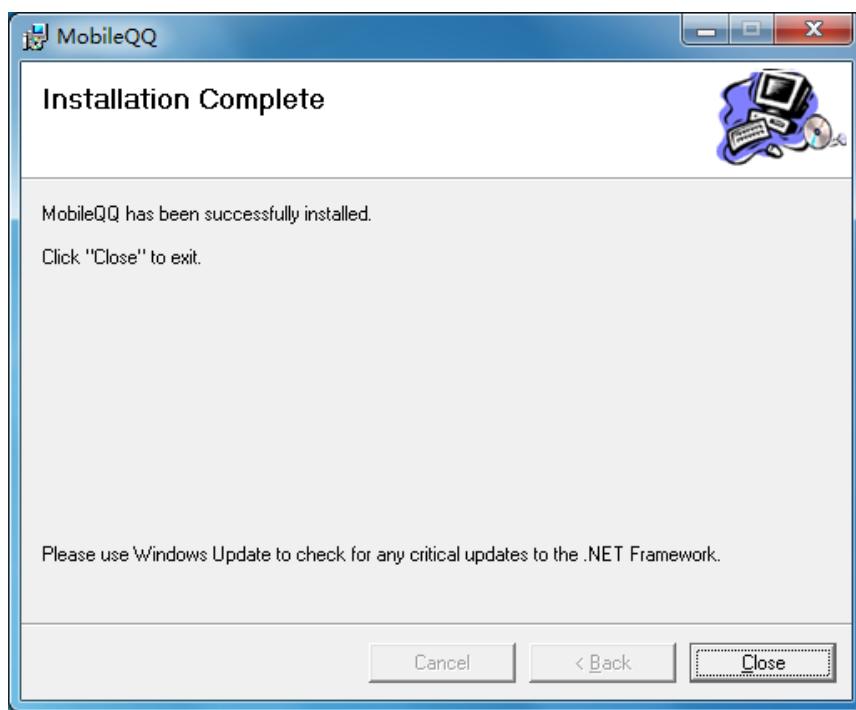
Step5: click on “Next” to continue



Step6: the installation is kicked off.



Step7: click on “Close” to complete

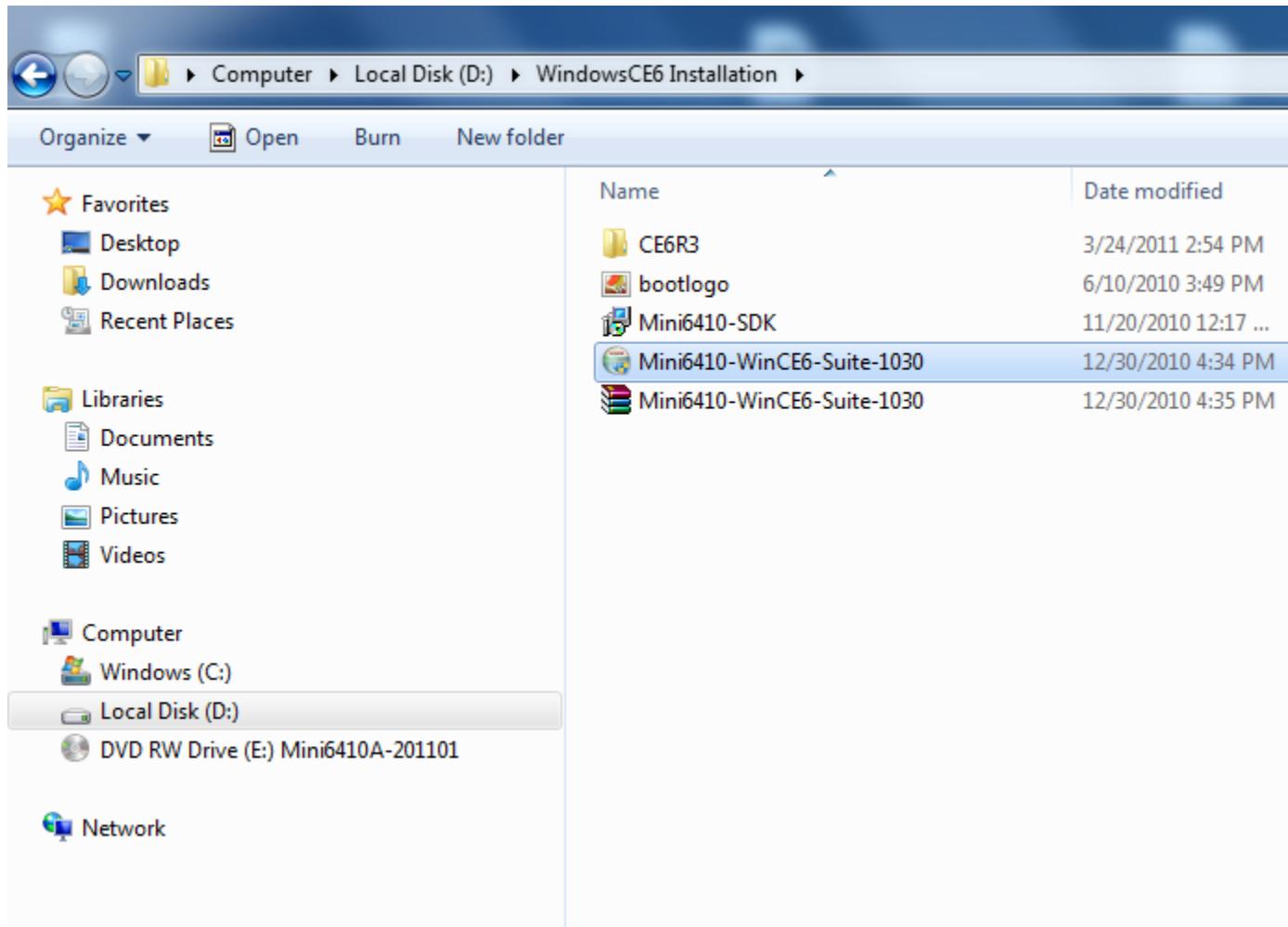


5.2.4 Install BSP and Sample Programs

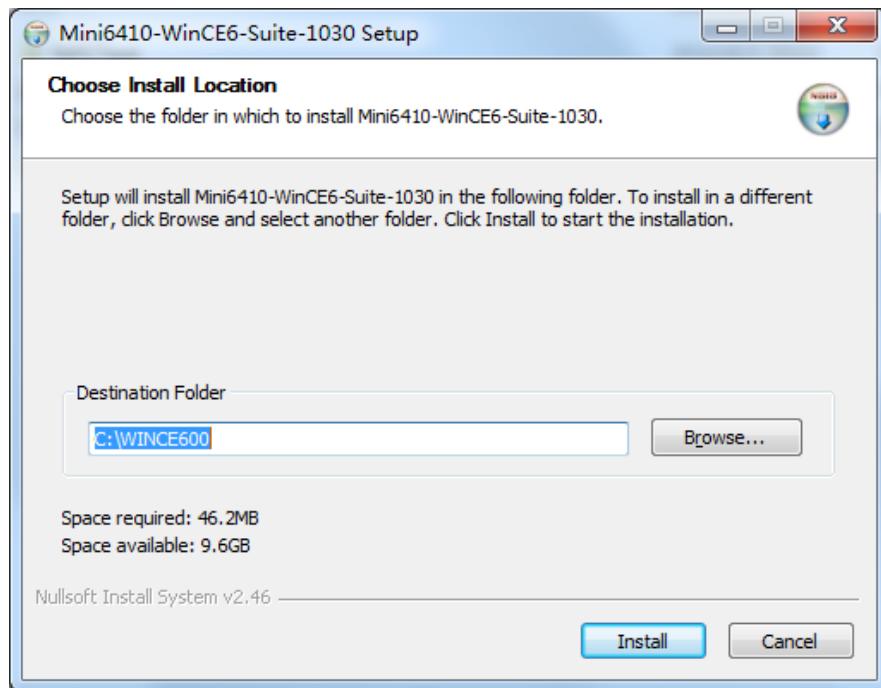
The Mini6410-1405 BSP and its project file can be installed by running

“Mini6410-WinCE6-Suite-1030” (1030 is the date when we released it. Our latest file one may have a different date, please refer to our CD). Users can download this file from <http://www.arm9.net>. Below are the steps to follow

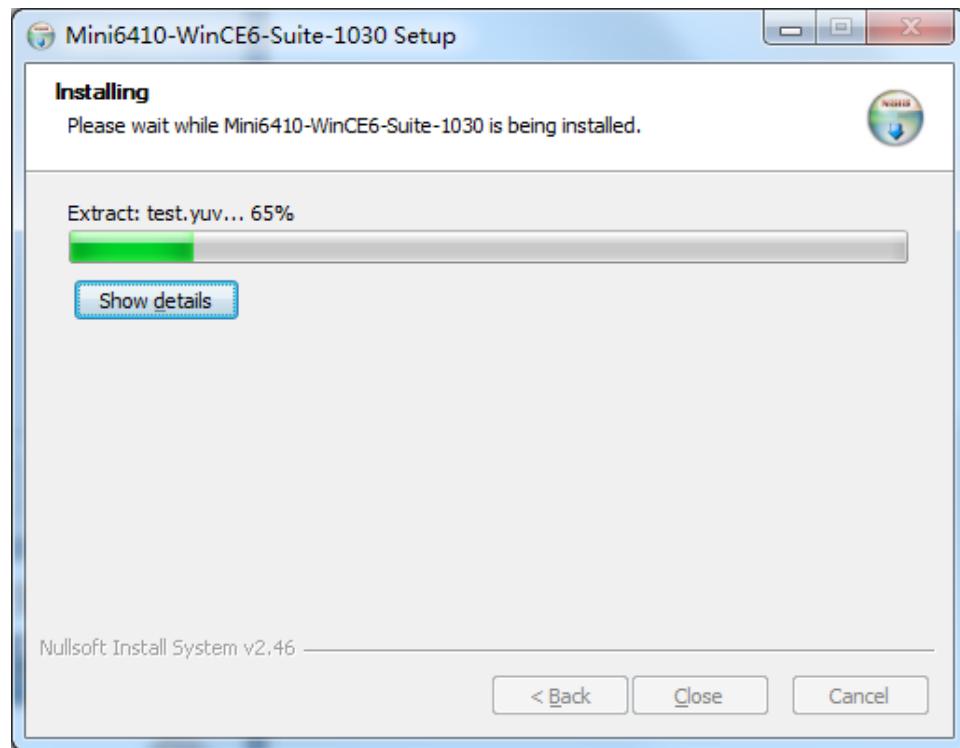
Step1: locate the “Mini6410-WinCE6-Suite-1030” executable and double click on it



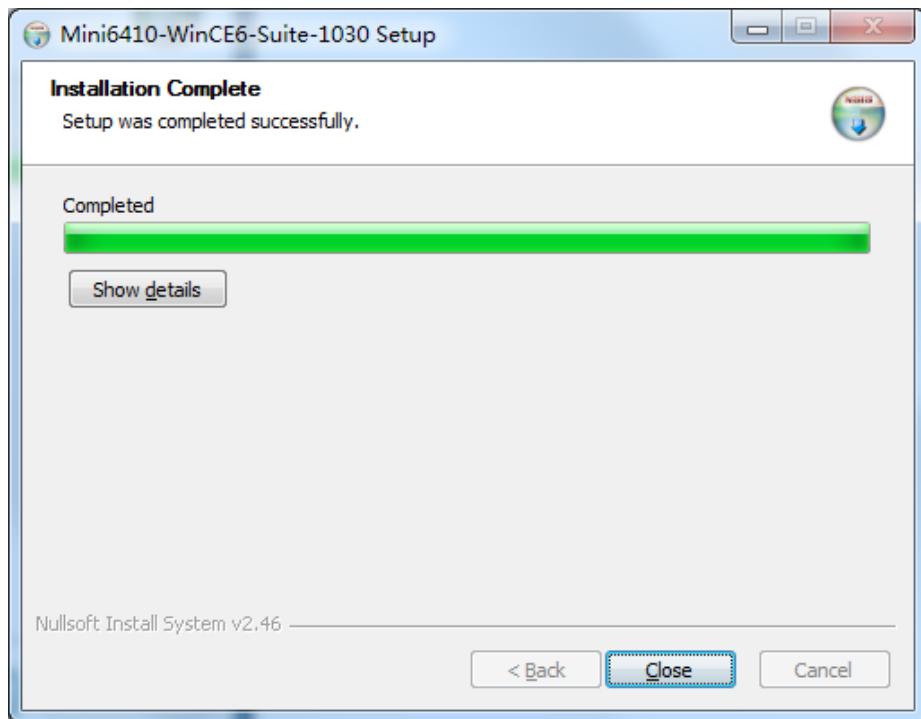
Step2: follow the default options and click on “Install” to continue



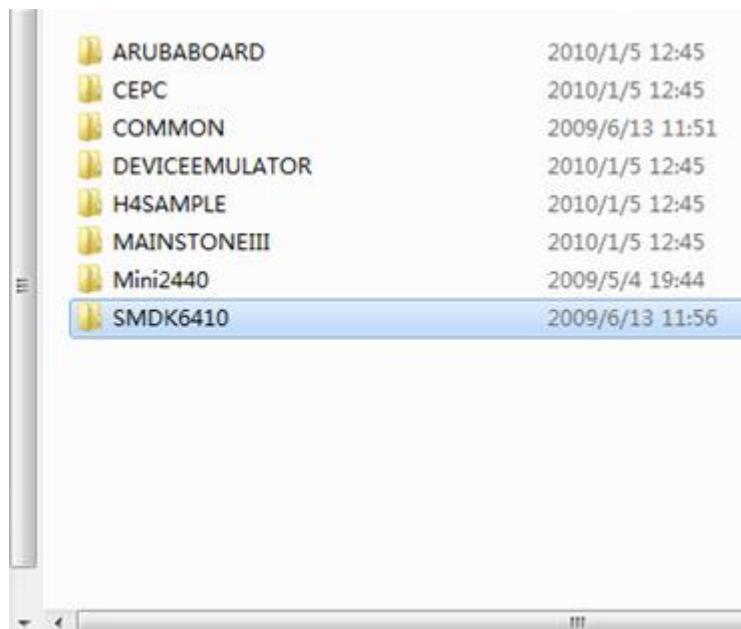
Step3: the installation is kicked off and very short since the file is very small



Step4: after the installation is done, click on “Close” to complete

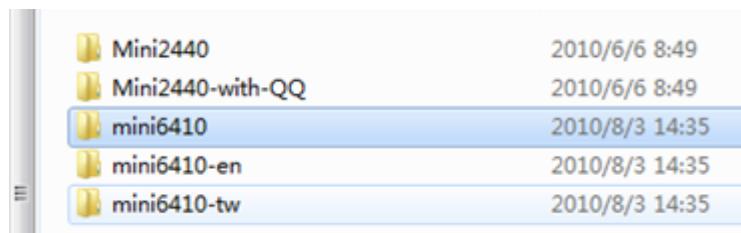


A SMDK6410 BSP directory will be created under “WinCE600\PLATFORM”



Three directories for three languages will be created under “WinCE600\OSDesigns”:

- Mini6410 - Simplified Chinese
- Mini6410-en - English
- Mini6410-tw – Traditional Chinese



Now you have successfully set up your Windows CE 6.0 development environment.

5.3 Configure and Compile WinCE6.0 Kernel and Bootloader

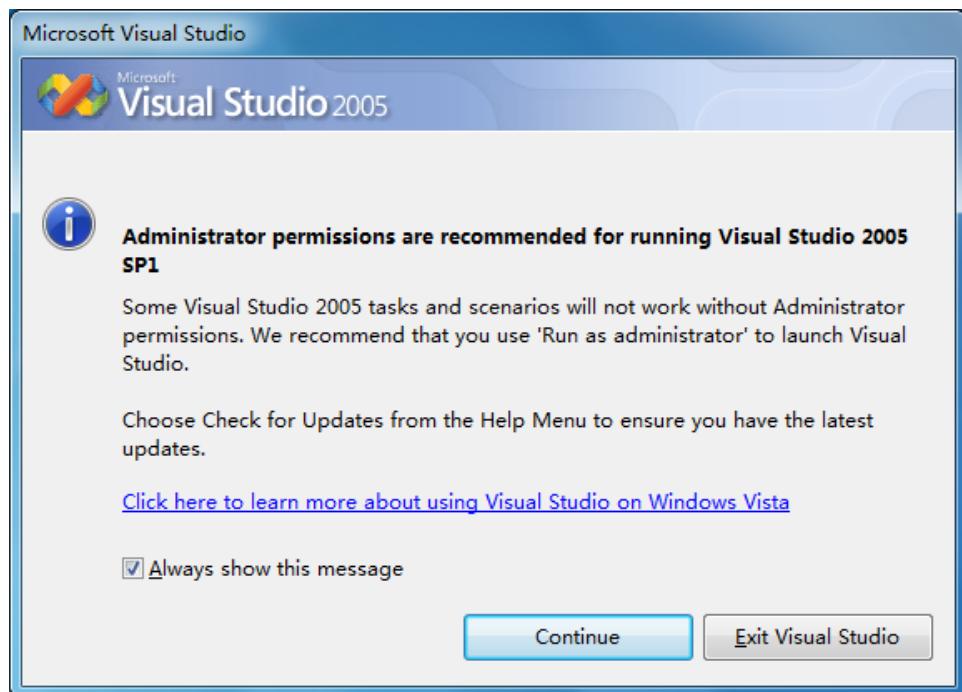
Configuring a Windows CE6 kernel is very complicated and users may easily fail if they don't do it carefully. Therefore we prepared a kernel project for users' reference. You can follow the steps below to open and compile it. There is a ready image under “images\WindowsCE6” in the shipped CD

5.3.1 Compile Kernel

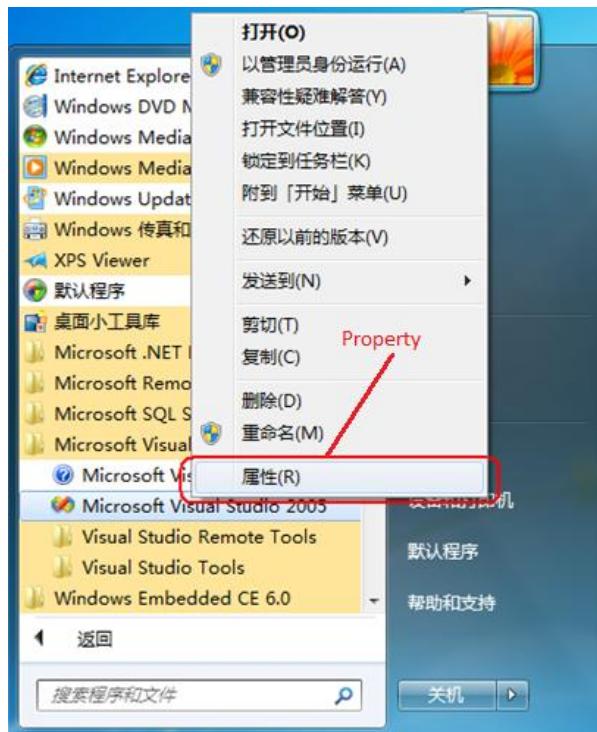
Please launch VS2005 to begin compiling the mini6410-1405 BSP.

Step1: go to “Start”->“Programs”->“Microsoft Visual Studio 2005”->“Microsoft Visual Studio 2005”

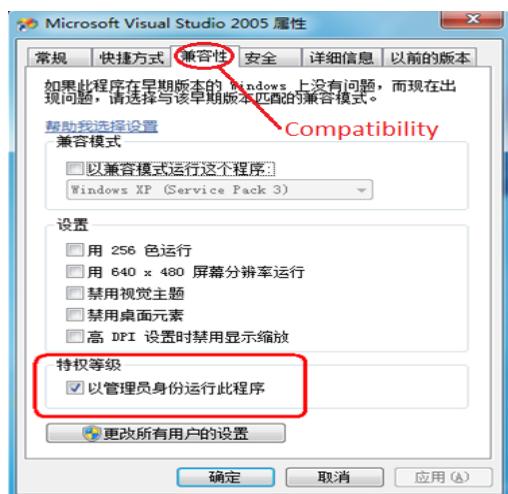
Step2: click on “Exit Visual Studio” on the following dialog



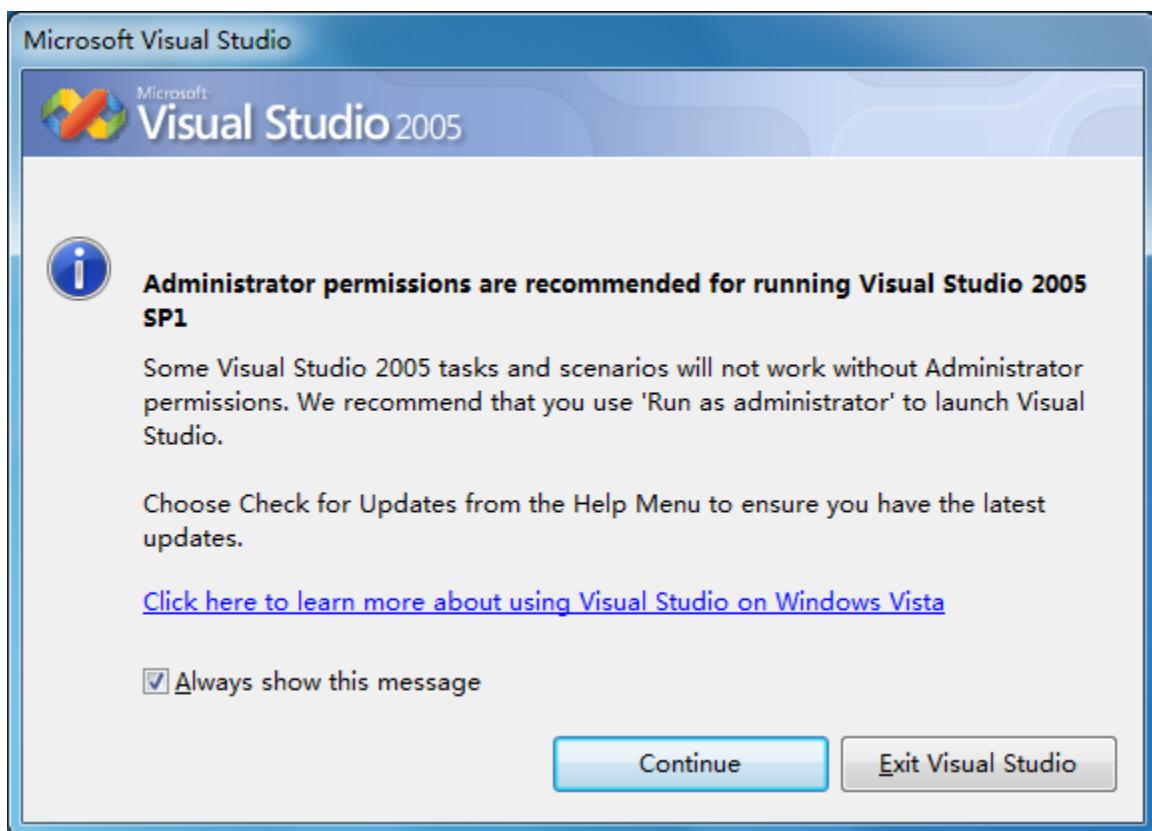
Step3: Go to “Start”->“Programs”->“VS2005”->“VS2005”, right click on it and you will see the following dialog. Select “Property”. We need to run it as administrator.



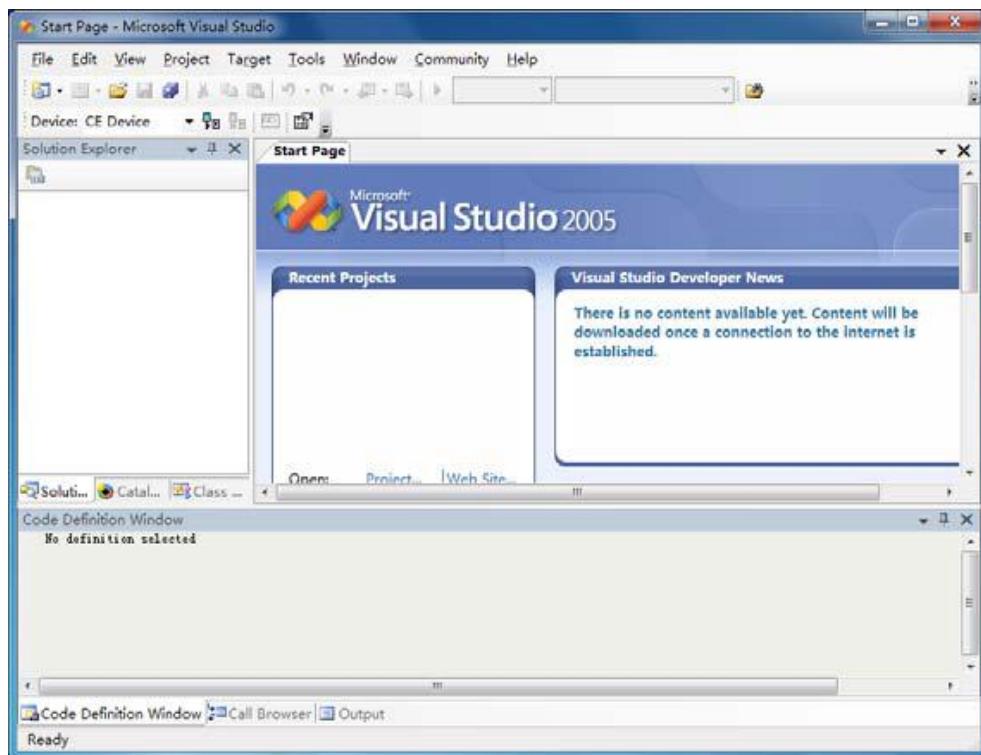
Step4: in the dialog shown below, click on “compatibility” and check the option shown and click on “OK”



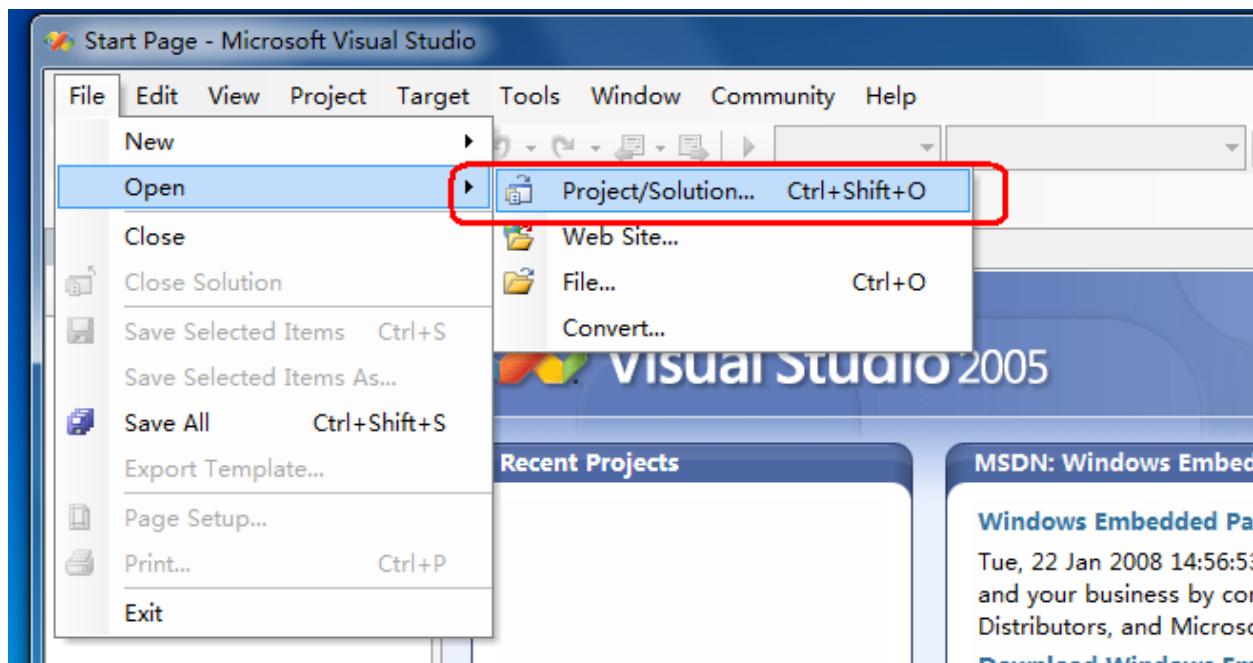
Step5: go to “Start”->“Programs”->“VS2005”->“VS2005” and click on “Continue” and you will run it as administrator



Step6: now you will see the initial interface of VS2005

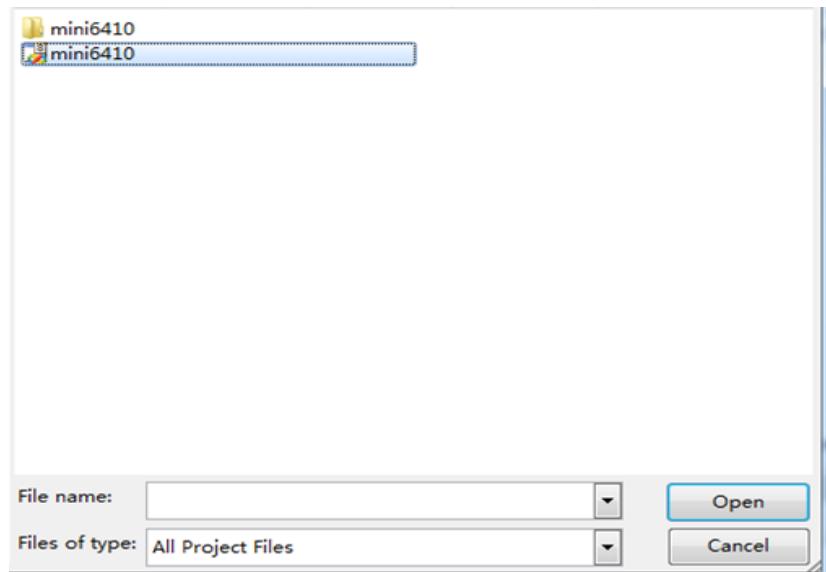


Step7: Go to “File->Open->Project/Solution...”

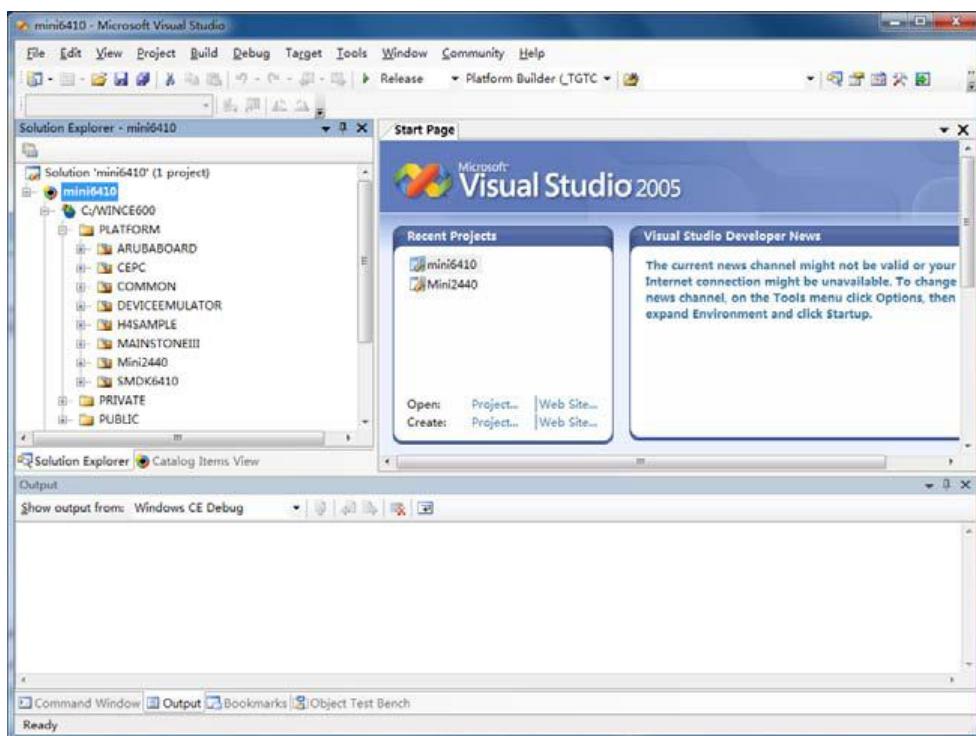


Step8: locate your mini6410 file (in this example it was

C:\WINCE600\OSDesigns\Mini6410), click on “Open”

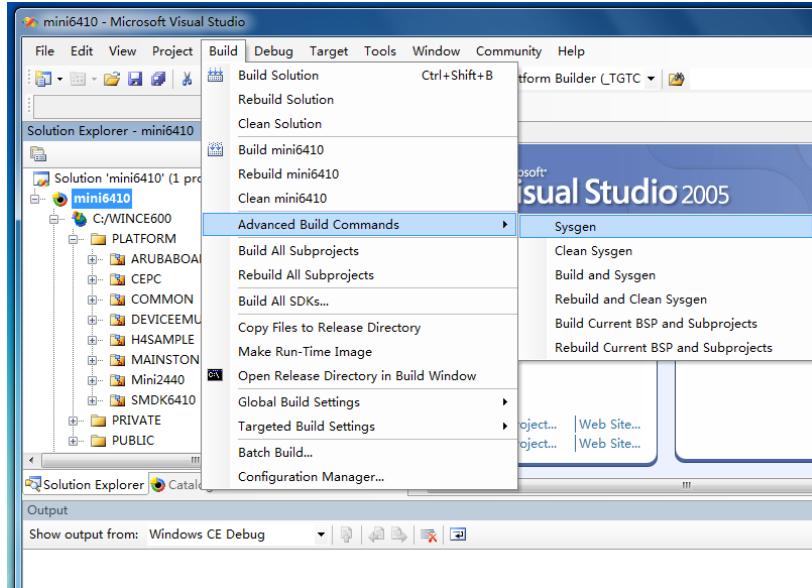


Step9: moments later after the mini6410 project is loaded you will see the following dialog



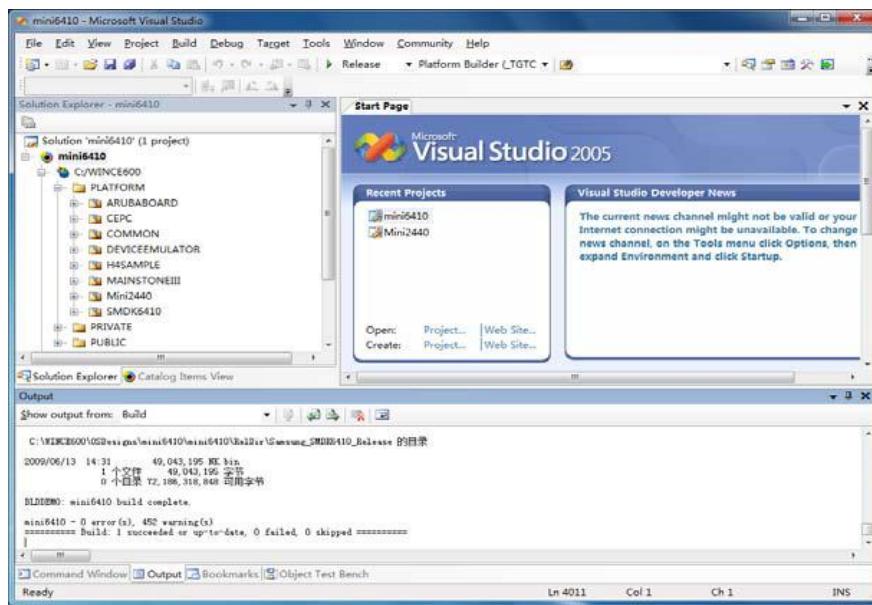
Step10: go to “Build->Advanced Build Commands->Clean Sysgen” to begin compilation.

This process may take a while



Step11: after the compilation is done, an NK.bin and an NK.nb0 will be generated under the following directory:

C:\WINCE600\OSDesigns\Mini6410\Mini6410\RelDir\Mini6410_ARMV4I_Release



5.3.2 Configure LCD Type and Serial Port in BSP

Our BSP currently supports the following LCD models:

- NEC 4.3”LCD with touch screen
- TPO 3.5”LCD with touch screen
- Inonux 7”LCD with touch screen
- Sharp 8”LCD (or compatible models) with touch screen
- LCD2VGA conversion module: 1024x768, 800x600 and 640x480
- EZVGA: a simple VGA conversion module, resolution 800x600

You can define your LCD type by setting up the LCD_TYPE value in “\SMDK6410\SRC\INC\options.h”:

```
#define LCD_N43 – for NEC4.3”LCD by default  
##define LCD_T35  
##define LCD_L80  
##define LCD_A70  
##define LCD_VGA1024768  
##define LCD_VGA800600  
##define LCD_VGA640480  
##define LCD_EZVGA
```

Users can modify the serial port function in “options.h” too:

```
// --- by customer
```

```
#define KITL_NONE - default setting
```

```
##define KITL_SERIAL_UART0
```

```
##define KITL_SERIAL_UART1
```

The default setting here is to make it a common serial port (there are some issues now to use COM1 as a common serial port). If you want to set it as an output for debugging information you can follow the changes below:

```
##define KITL_NONE
```

```
#define KITL_SERIAL_UART0
```

```
##define KITL_SERIAL_UART1
```

5.3.3 One Wire Precise Touching

To facilitate touch screens we especially designed a circuit for accurate touching for the Mini6410-1405 system. It utilizes the ADS7843 (or compatible models) chip and incorporates a single chip machine to form an independent four wire resistor data collection circuit. It can collect accurate data and filter noises and send data to a common GPIO. The PWM1 on the board is connected to it and actually we only use GPF15. The driver for the accurate touching function has been compiled as a dll(touch_1wire.dll) and integrated into our BSP. But before you can enable the function you need to configure it and compile the kernel. Below are the steps to configure it.

Open “C:\WINCE600\PLATFORM\SMDK6410\ SMDK6410.bat”, locate the following

lines:

```
set BSP_NOTOUCH=
set BSP_NOTOUCH_ADC=1
set BSP_NOTOUCHCOM=1
set BSP_NOTOUCH_1WIRE=
```

If an item's value is null it means the system supports it otherwise the system doesn't support it therefore if you

want to utilize the touch screen control that comes with ARM you can do it this way:

```
set BSP_NOTOUCH=
set BSP_NOTOUCH_ADC=
set BSP_NOTOUCHCOM=1
set BSP_NOTOUCH_1WIRE=1
```

If you want to use the serial port control touch screen you need to make changes as below:

```
set BSP_NOTOUCH=
set BSP_NOTOUCH_ADC=1
set BSP_NOTOUCHCOM=
set BSP_NOTOUCH_1WIRE=1
```

By default our system is configured to support accurate touching. To differ this kind of image from the one that doesn't support accurate touching we append “-i” to its file name such as “NK_A70-i.bin” in the shipped CD. Or you can compile your own kernel that

supports the serial port touch screen. Previous we differed these two by appending “-s” to its file name.

To test your touch screen you can launch the “Painter” utility in the system. You will find it moves very smoothly without vibrations



5.3.4 Bootloader

In the Mini2440 system the bootloader for WindowsCE5/6 is nboot which is compiled with ADS. In the Mini6410-1405 system we name the bootloader “nboot” too and its source code is together with the BSP. It needs to be compiled with VS2005.

The source code of nboot is under

“C:\WINCE600\PLATFORM\SMDK6410\SRC\BOOTLOADER”

This directory includes the following two nboot files:

- nbootRAM128: for 128M RAM system

- nbootRAM256: for 256M RAM system

The two nboot will be compiled together.

As far as functions are concerned the nboot for the Mini6410-1405 is the same as the one for the Mini2440. They are simple and less than 8K(the Mini2440's nboot is less than 4K). In general the bootloader will be burned into the NAND Flash's Block 0 to boot the WinCE kernel. Originally the nboot was from Samsung and we made some improvements. Our version has the following features:

- Support boot logo
- Support process bar for WinCE kernel loading
- Rapid WinCE Booting

Nboot doesn't support file burning and can only read files: BootLogo and the WinCE.

Nboot is easy to be customized. You can set its bootlogo' position, background and process bar's color, length and width in “**options.h**”. This file is for the BSP and under “**SMDK6410\SRC\INC**”.

```
#define KITL_NONE

##define KITL_SERIAL_UART0

##define KITL_SERIAL_UART1

#define LCD type, the default is NEC 4.3"LCD

#define LCD_N43

#define LCD_T35
```

```
##define LCD_L80
##define LCD_A70
##define LCD_VGA1024768
##define TOUCH_SCREEN_WIDTH 1024
##define TOUCH_SCREEN_HEIGHT 768
//define background color
#define BACKGROUND_R 0x00
#define BACKGROUND_G 0x00
#define BACKGROUND_B 0x7F
//define process bar's color
#define PROGRESS_BAR_R 0xFF
#define PROGRESS_BAR_G 0xFF
#define PROGRESS_BAR_B 0x00
//define bootlogo's position
#define LOGO_POS_TOP 0
#define LOGO_POS_LEFT 0
//define process bar's position, length and width
#define PROGRESS_BAR_TOP 240
#define PROGRESS_BAR_LEFT 50
#define PROGRESS_BAR_WIDTH 400
```

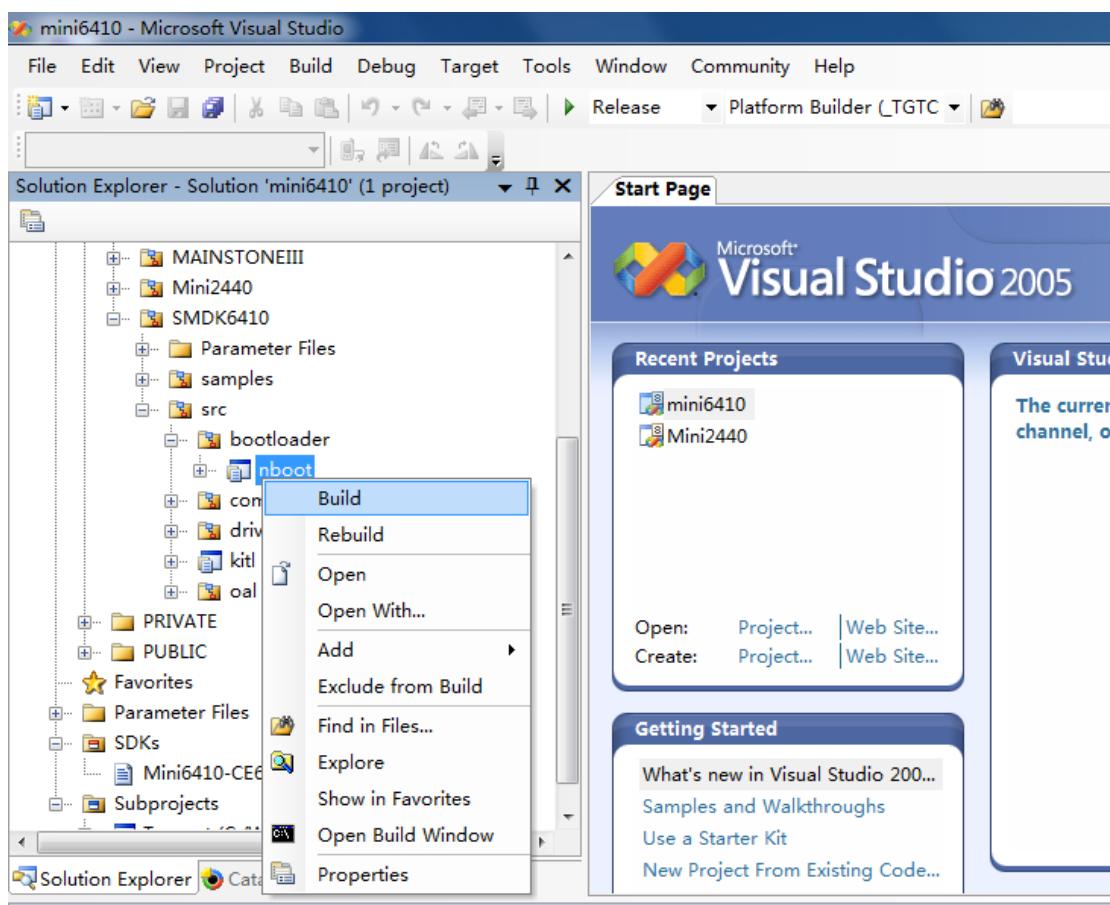
```
#define PROGRESS_BAR_HEIGHT 4
```

Compiling Nboot

Actually in our previous steps a Nboot has been compiled and an object file “Nboot.nb0” is ready for use. Its format is the same as the one compiled with ADS. We now need to burn it into the board. It is located under

“C:\WINCE600\OSDesigns\mini6410\mini6410\RelDir\Samsung_SMDK6410_Release”.

Compiling a whole WinCE kernel takes a long time. You can just compile the nboot alone. In the browser locate your nboot source code directory, right click on it and select “Build”.



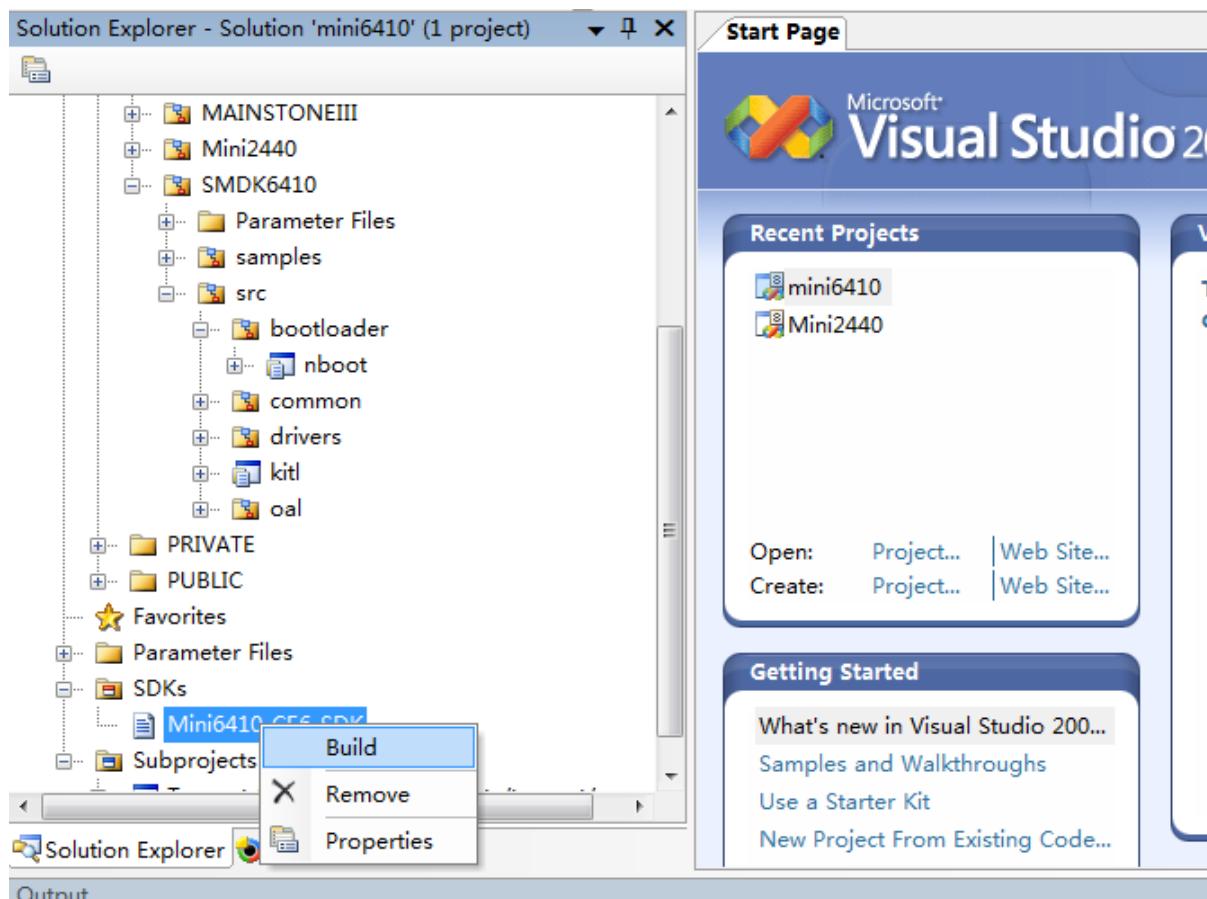
Now you need to burn your nboot.nb0 to the NAND flash

5.3.5 Create SDK

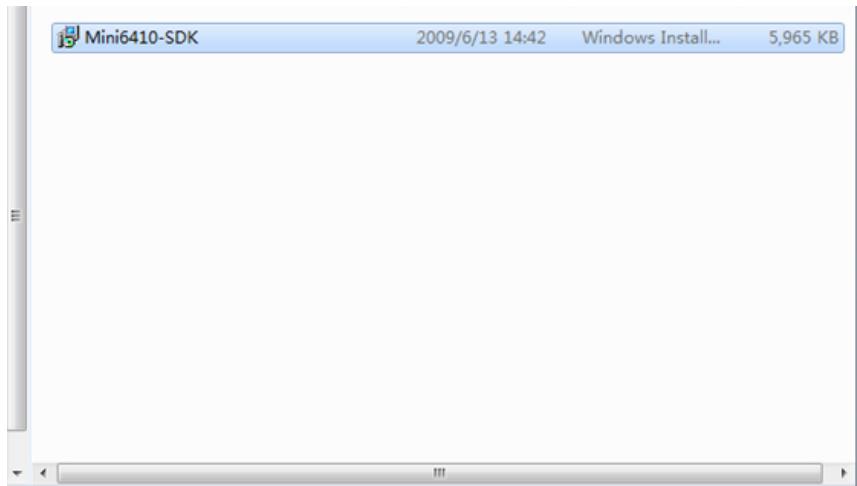
After you installed VS2005 on your PC but didn't install Windows CE 6.0 Platform Builder you must have a SDK to develop MINI64140 applications with VS2005. It is very similar to the SDK needed for Embedded Visual C++.

After you compile your kernel you can set up a SDK via VS2005. Note: this SDK is only for VS2005 not for other utilities such as EVC and VS2008. Please follow the steps below to do it:

Step1: start VS2005, open your project file mini6410, right click on “Mini6410-CE6-SDK” and select “Build”



Step2: moments later the SDK will be built



Step3: go to “C:\WINCE600\OSDesigns\mini6410\mini6410\SDKs\mini6410sdk” you will be able to find a “Mini6410-CE6-SDK.msi” installation file

5.3.6 Install SDK

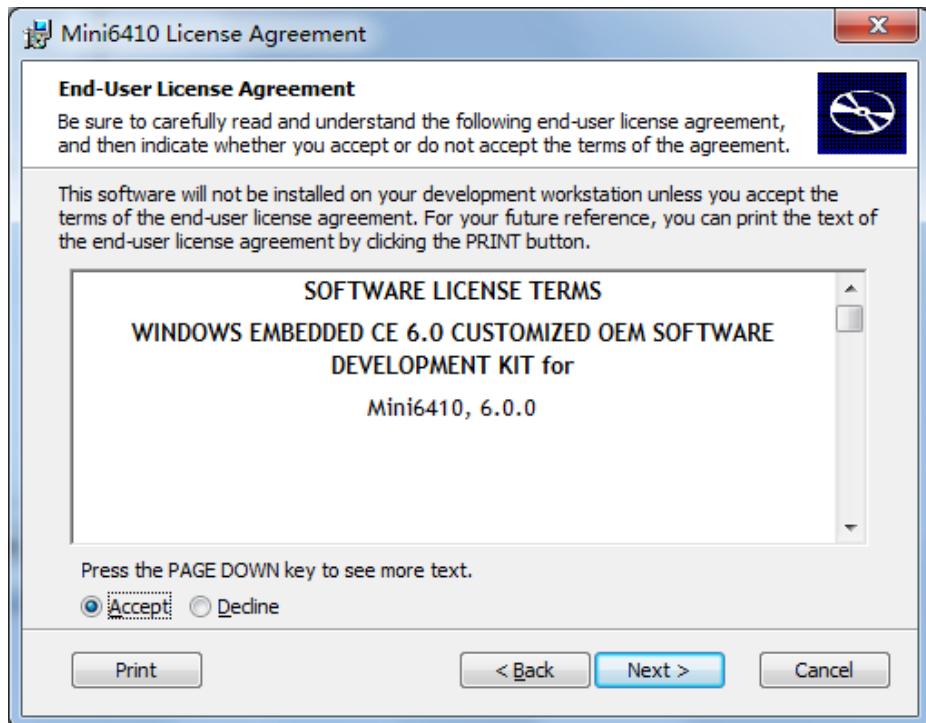
Note: our shipped CD already has an SDK under “WindowsCE6\Mini6410-SDK.msi”.

Now let's install an SDK

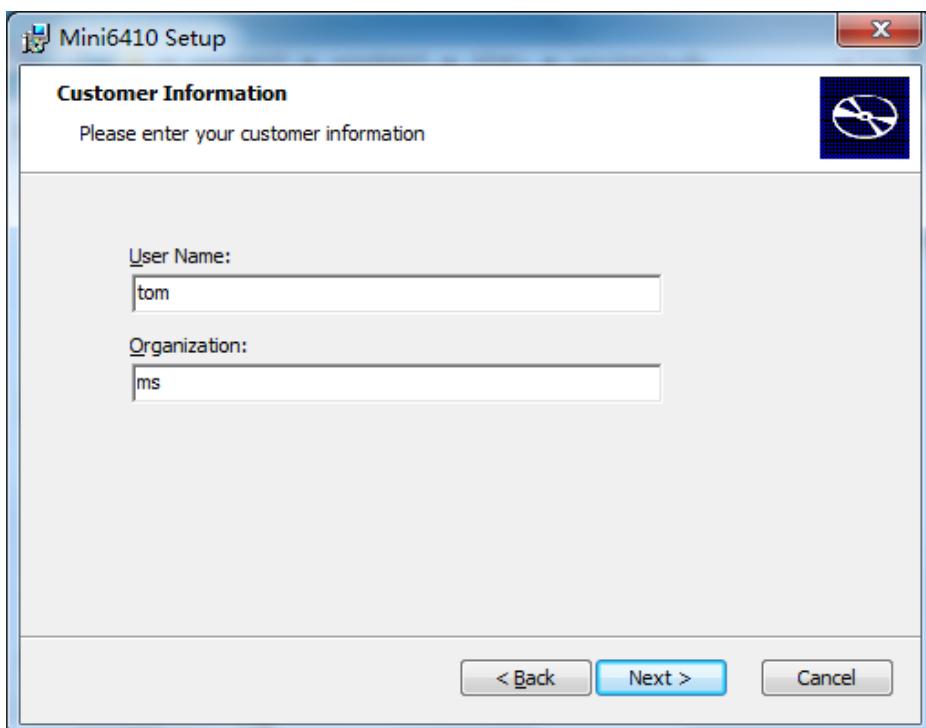
Step1: double click on “Mini6410- SDK.msi” and click on “Next” to continue



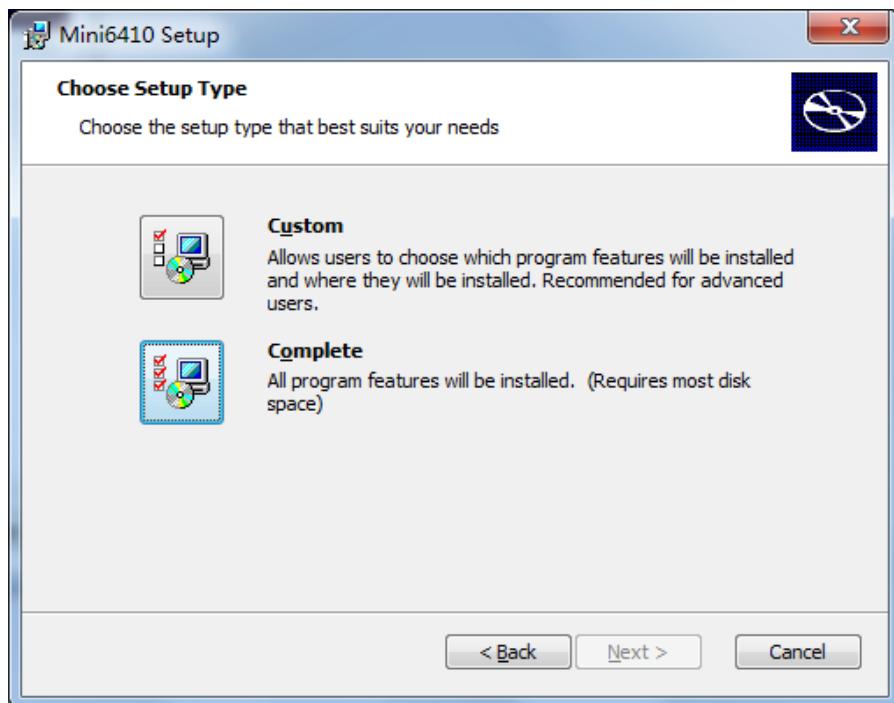
Step2: check “I accept” and click on “Next” to continue



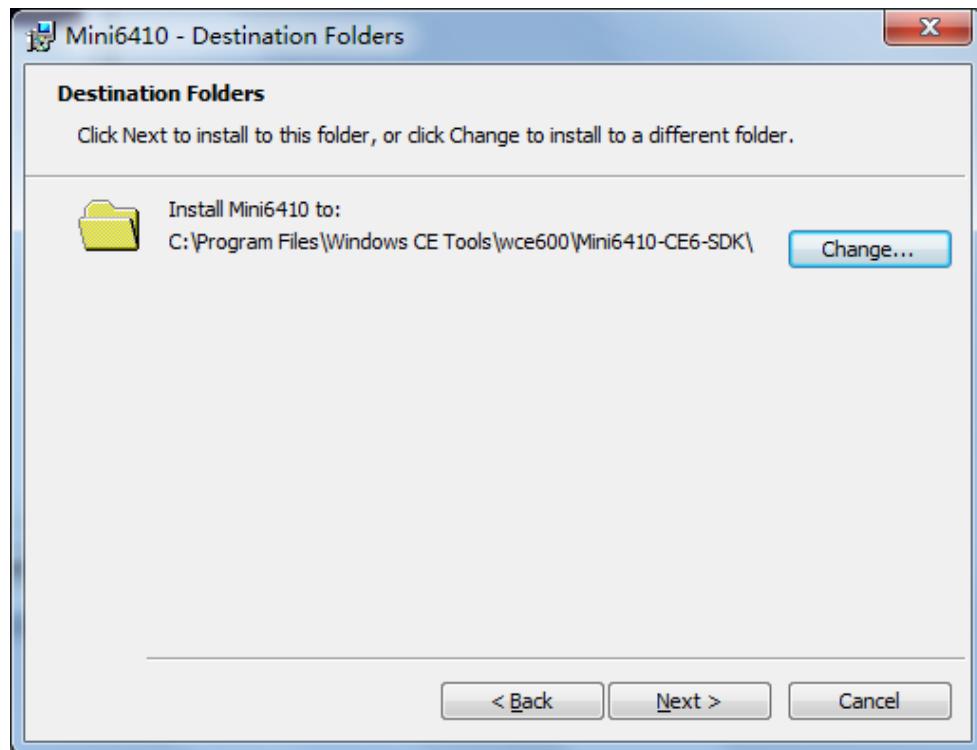
Step3: type your user name and company's name and click on “Next” to continue



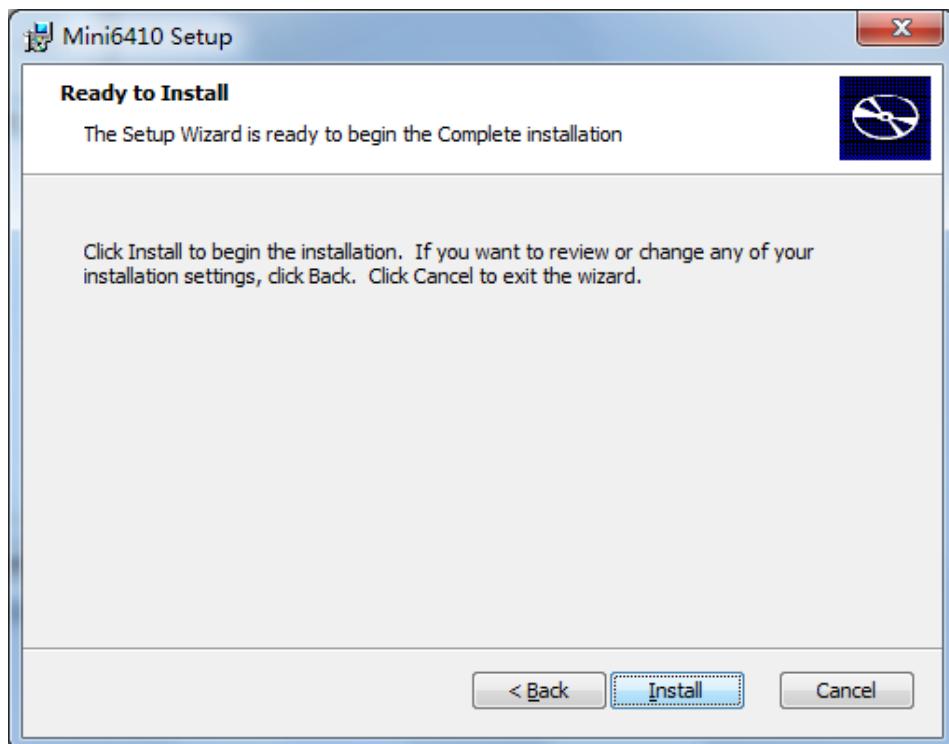
Step4: click on “Complete” to continue



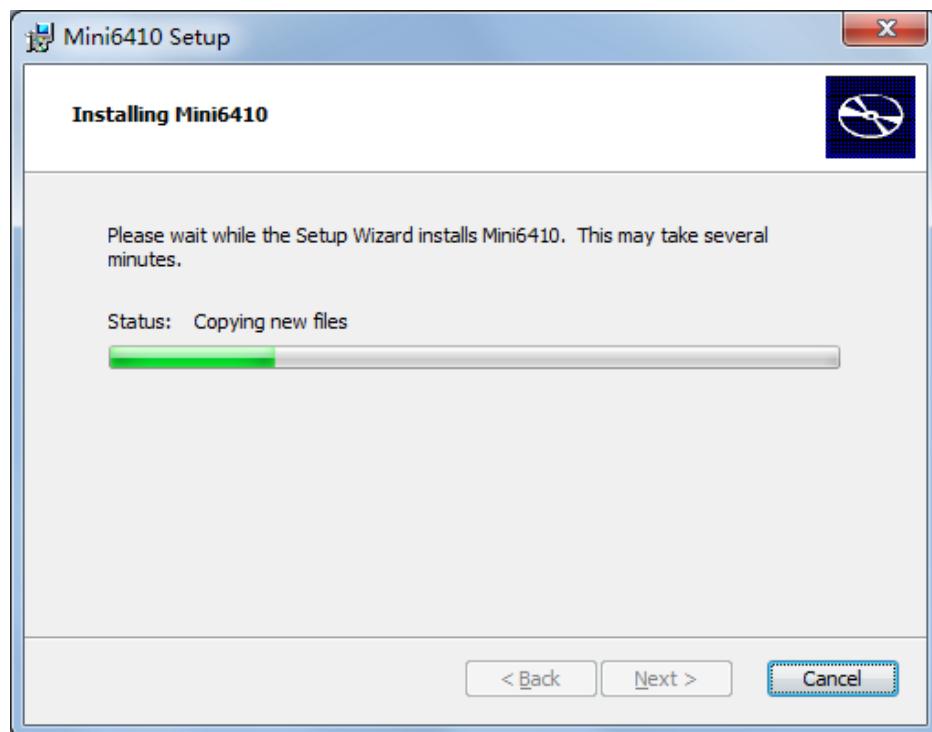
Step5: click on “Next” to continue



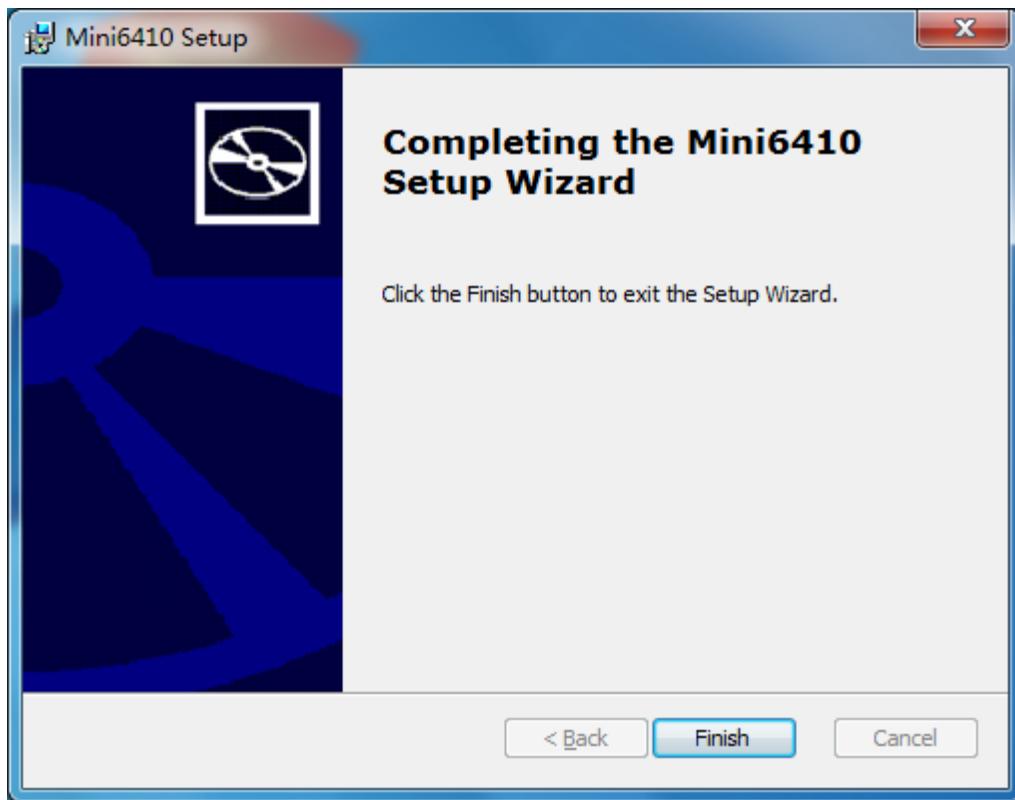
Step6: click on “Install” to continue



Step7: the installation is kicked off



Step8: click on “Finish” to complete



Now the SDK has been installed successfully

Chapter 6 Explore Android

The Mini6410-1405 system has enabled almost all and the latest features of Android such as 3G networking, USB Bluetooth, flash drive auto mounting and Ethernet setting. The 3G networking feature detects a USB network card automatically and supports all three systems: WCDMA, CDMA2000 and TD-SCDMA.

The version we used when we compiled this manual is Android-2.3.2

The bootloader that Android uses is very similar to the one for Linux. The only differences lie on the configurations. Actually Andorid's file system makes it special and when we talk about the Andorid system we are talking about its file system

6.1 Get Started with Android

6.1.1 Install Android

Note: running an ext3 Android system from the SD card on a 128M board may not be smooth or can even fail. We suggest our users run it on a 256M system from the NAND flash.

You can burn an Android system to your board via USB download or SD card or just run it from your SD card as follows

Step1: burn a Superboot to your SD card with SD-Flasher.exe

Step2: copy the whole image directory in the shipped CD to your SD card

Step3: open “\images\FriendlyARM.ini” in the SD card make these changes: “Action=Run”

and “OS=Android”

Step4: toggle the S2 switch to SDBOOT, insert your SD card, power on and your Android will be loaded

On your first system boot a calibration screen will pop up, please follow the “+” to calibrate your screen.

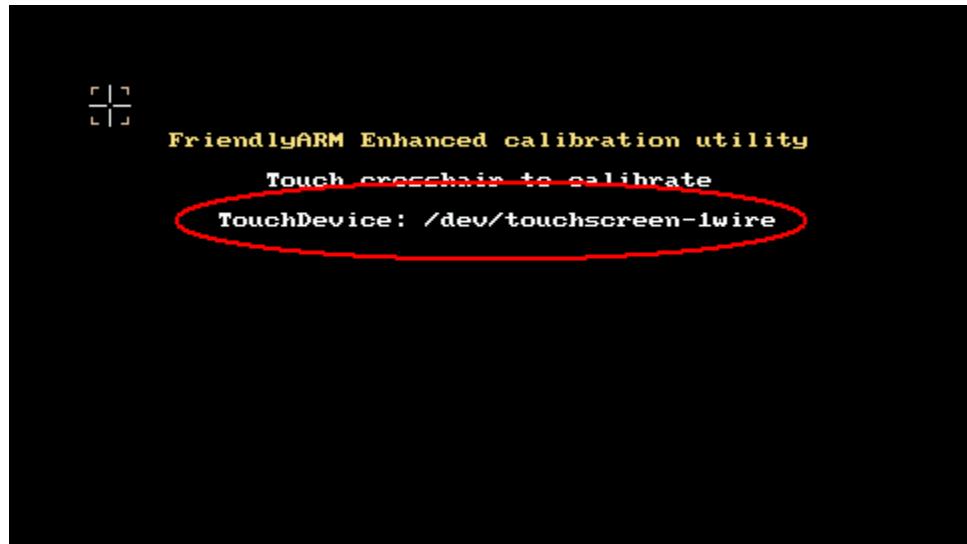
Note: some boards utilize a simulator to load the file system first and the first time loading of Android is handled by the simulator. This way runs Android a little bit faster. Our system uses a script to load Android. This way ensures data integrity and makes the process transparent, easy and convenient for users

Note: running Android from the SD card only utilizes two files:

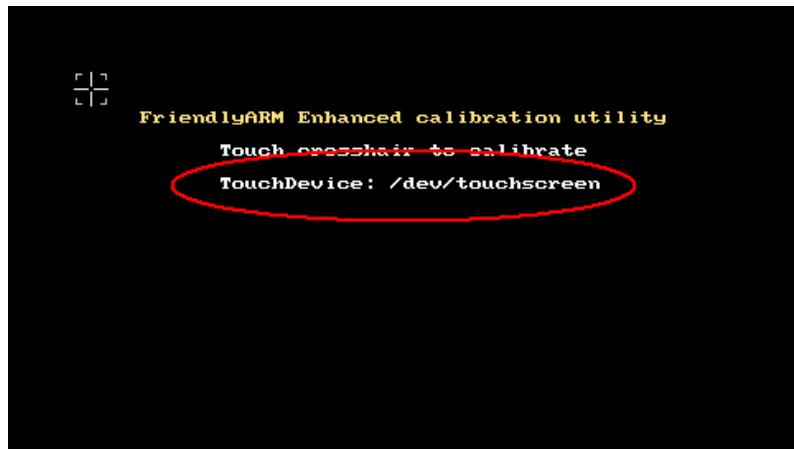
- azImage: kernel image. Different LCD systems require different images
- rootfs_android.ext3: EXT3 file system

6.1.2 Calibrate Touch Screen

After you burn an Android into your board you will see a calibration screen on the very first system boot. The following screen shows the system uses an accurate touching LCD (marked in red).



The following screen shows the system uses an ARM LCD (marked in red).



Follow the prompt, click on “+” to calibrate and you will enter the system after your calibration is done. If you don't position your pen properly the calibration process will restart until you are done successfully

6.1.3 Rotate Touch Screen

After Android 2.3 is loaded by default it will display vertically. To switch to a horizontal

screen please press the menu key(k2) and keep it down for a while it will change.



6.1.4 Icons on the Status Bar

We added four shortcut icons on the status bar in Android2.3, which allow users to easily operate the system from the touch screen



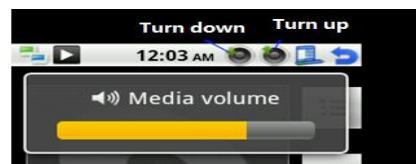
6.1.5 Play MP3

Android can detect MP3 files in the SD card. When you play a MP3 occasionally you may not hear any sound. In this case you can pause it and resume. This is an issue which is still troubling us. We are fixing this issue.



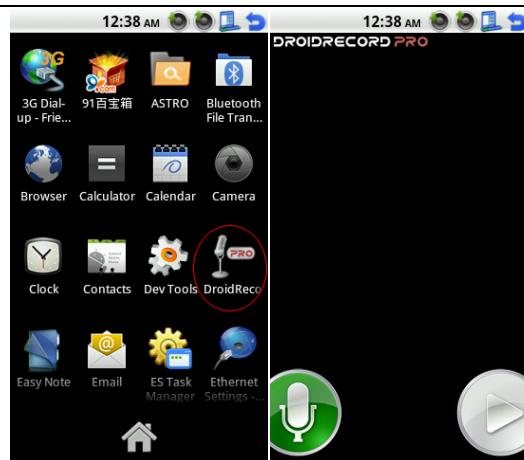
6.1.6 Adjust Volumn

When playing your audio you can adjust the volumn by clicking on the two speaker icons to turn it up or down



6.1.7 Audio Recording

The DroidRecord utility can record and play audio. Double click on the icon to launch it

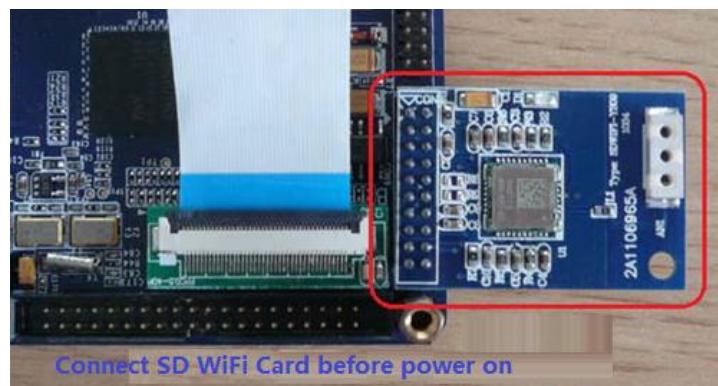


Please follow the screenshots below to start recording and play

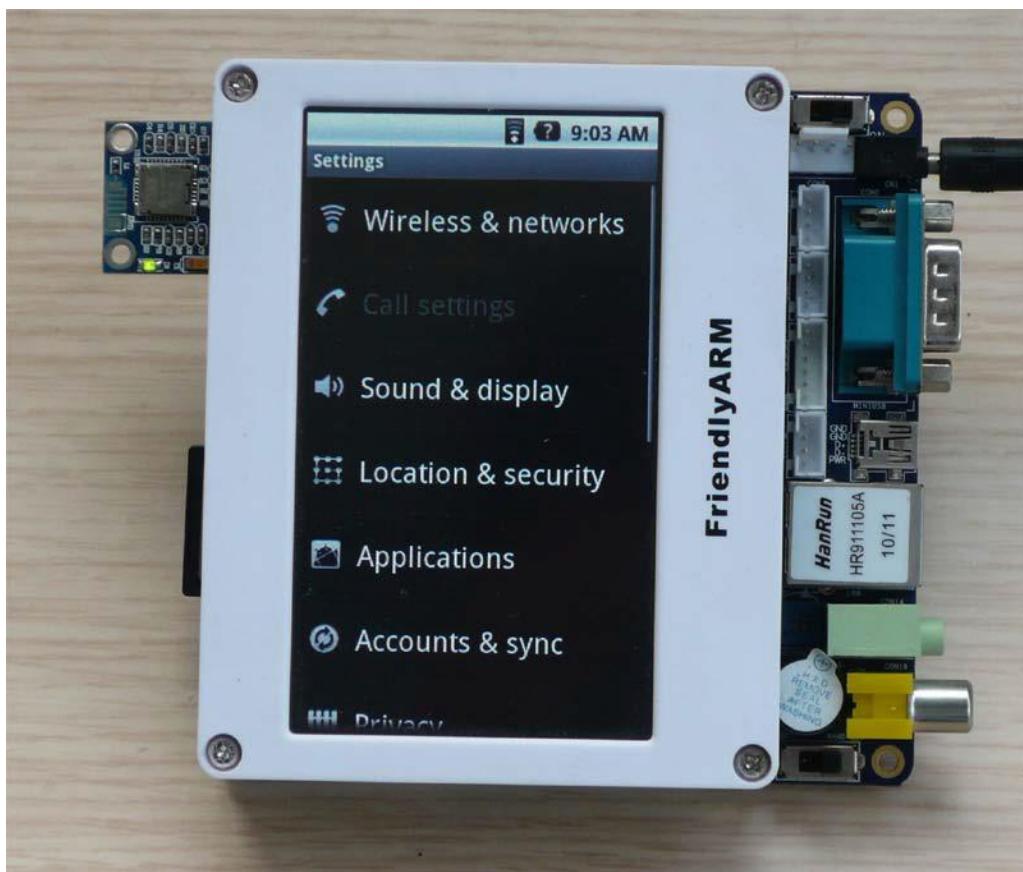


6.1.8 SD WiFi

Before power on your system please connect your SD WiFi to your board's SDIO (CON9)



Power on, press the Menu key (K2) click on “Setting” you will be able to see the following menu



Click on “Wireless & network” -> “Wi-Fi” to start the SD WiFi function and you will see that it is checked



Click on “Wi-Fi settings” and the system will search for nearby networks



Select your network and type required information to connect



Connection is successful



Click on “Home” to return to the Android main menu. Start a browser, type a website and



you will be able to visit it



6.1.9 USB Camera

The USB camera is plug and play. Please click on the “Camera” icon.



You can preview with your camera



6.1.11 GPS

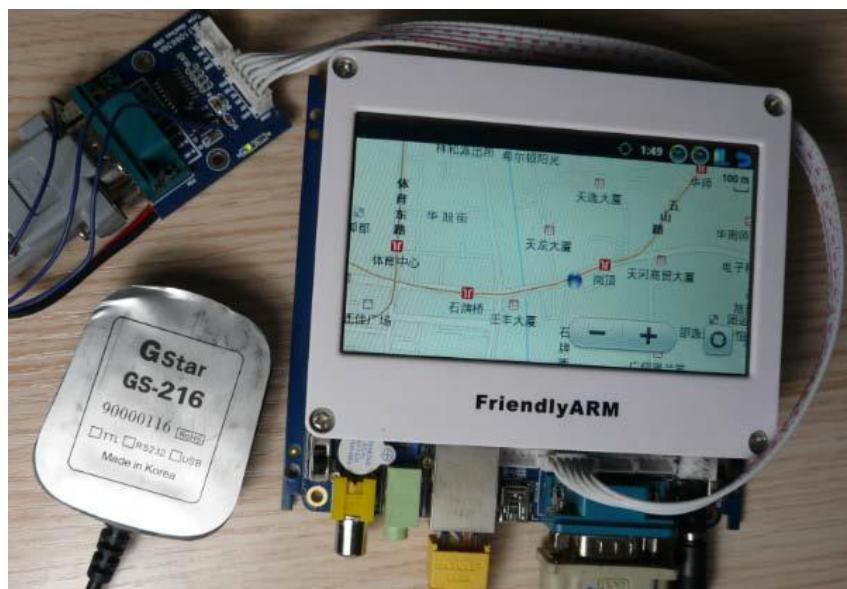
Our Android 2.3.4 supports both serial port and USB GPS devices. If you use a USB device please set the device to “/dev/ttyUSB0” in the “init.rc” file e.g. “setprop ro.kernel.android.gps/dev/ttyUSB0”. By default the device is “/dev/s3c2410_serial1”

We will present how to connect a serial port GPS in detail

Please connect your GPS to your board as follows



Then connect your board to the internet and open a map (e.g. maps.google.com). Minutes later you will find your location.



Note: please test this outdoor

6.1.12 Configure Ethernet

Android has an ethernet configuration utility



Click on it you will see the following dialog



1. Configuring Network Parameters Manually

Click on “Close” and then “Setting” to configure the network parameters



You will see that “Ethernet Network” is checked it means the Ethernet is working. This is the same as “ifconfig eth0 up”.

“Use static IP” is checked too and this means you need to configure the network manually.

Click on “IP address” and you will see the following dialog. Please type your network information and click on “OK” to save



Please type other information as well such as Gateway, Netmask and DNS

After you are done please press K1 to return to the previous interface then you will see the following dialog



Click on the icon you will see the current network information

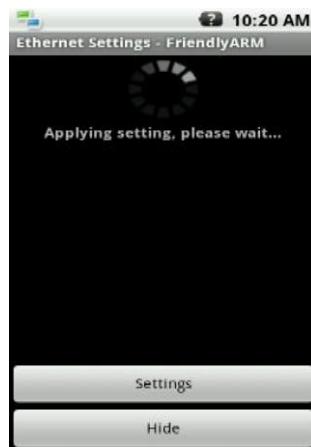


2. Auto Configuring IP with DHCP

Uncheck "Use static IP" you will see the following screenshot



Using DHCP doesn't allocate DNS automatically. You still need to set it. After you are done press K1 you will see the following dialog



If everything is OK you will see the following screenshot



Now you can surf the internet



6.1.13 3G

We specially developed a 3G network utility for Android. It can automatically detect and support up to more than one hundred USB network cards for all these systems: WCDMA, CDMA2000 and TD-SCDMA. We have a list of the USB 3G cards that are supported in 4.1.21.

Our following example was tested with HUAWEI E1750 for WCDMA

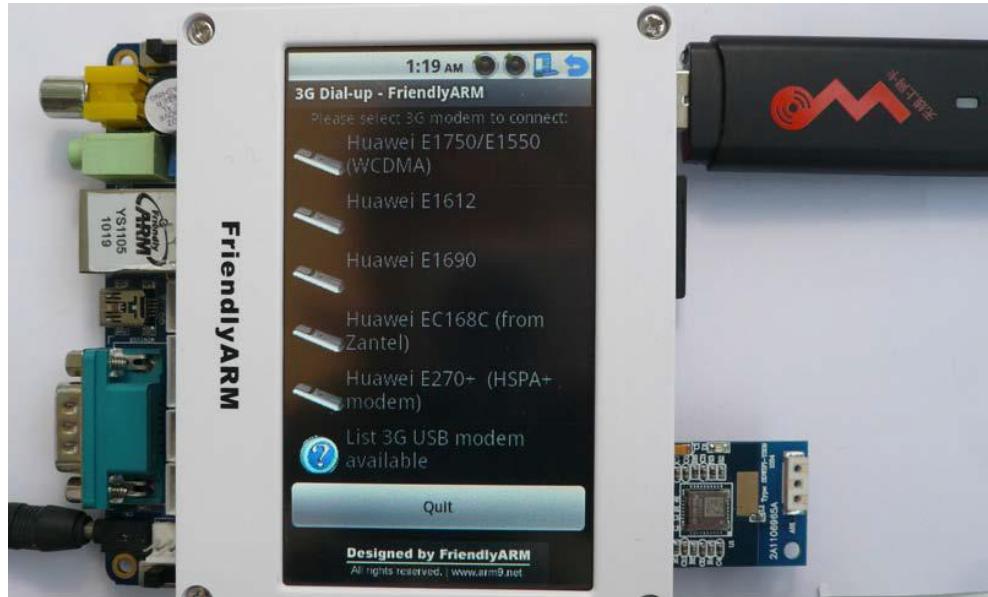
Step1: Insert a SIM card into your USB card



Step2: Connect your USB card to the board and start the 3G utility



Step3: the 3G utility will detect the E1750 card. Click on its icon



Step4: in the dialog shown below there is an orange icon with a “-” in the center. This means no network is connected. Click on “Connect” to start connection



Step5: the connection process may take a while



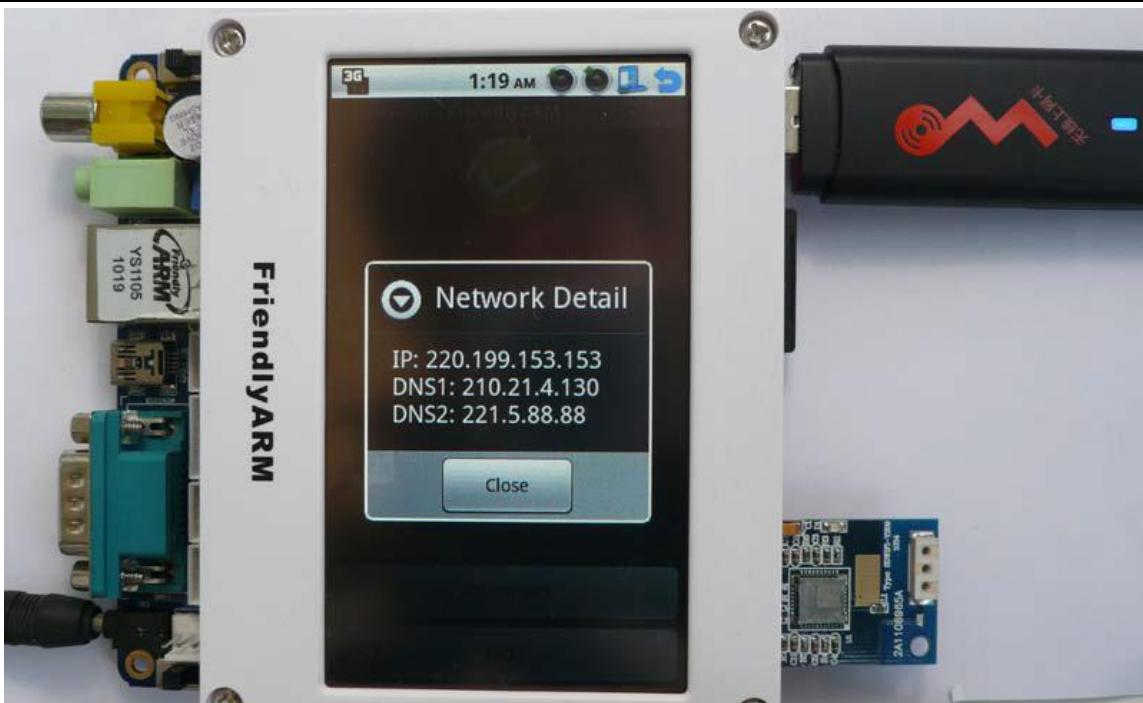
Step6: if the connection is a success the orange icon will turn green and shows “Connected” and meanwhile FriendlyARM’s websites will be listed and a “3G” icon will show up in the upper left of the screen



Step7 Click on the green icon you will see the current network information



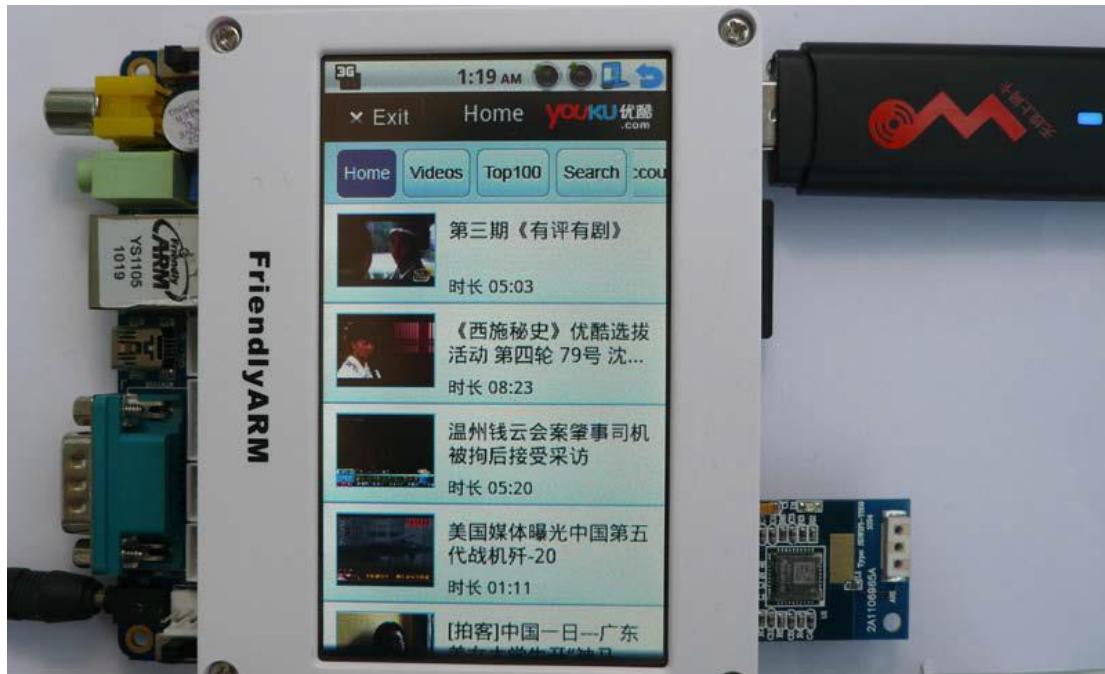
Complete ARM Solutions
Design, Development and Manufacturing
Expertise on Embedded Linux, Android, WindowsCE



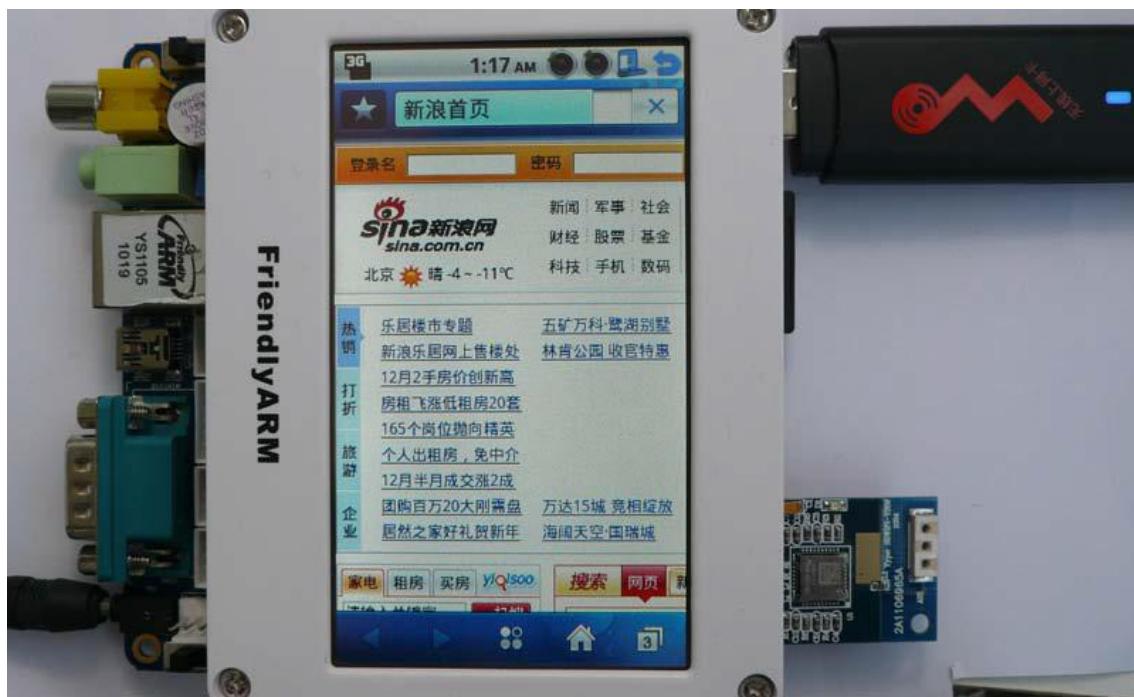
Step8 you can click on “Hide” to run it on background



Step9 try youku.com



Try QQ browser:



Step10: to close the connection click on the “3G Network Status” icon to return to the main menu and click on “Disconnect”



6.1.14 USB Bluetooth

Android supports various USB bluetooth adapters. Please connect your USB Bluetooth card to the USB host on the board



Press K2 and click on “Settings” to enter the configuration menu



Click on “Wireless & networks” to enter the wireless network setting



Click on “Bluetooth settings” and check “Bluetooth” to start the Bluetooth service. It will search for nearby bluetooth devices and list them



6.1.14.1 Bluetooth Communication

Please get a cell phone which supports bluetooth and start the Bluetooth service. Boot your board with Android, go to “Bluetooth settings”, click on “Scan for devices” and it will find your cell phone (in our example it was “A760 BT”)



Click on the cell phone name, type the password and click on “OK”



At the same time there is a dialog shown on your cell phone prompting you to input a password. Type the same one you did on the board.

If the connection is a success, on the “Bluetooth settings” interface you will see “Paired but not connected” under your cell phone name.



6.1.14.2 Transfer File to Cell Phone

Please follow the steps described in the previous section to connect your board to a cell phone. On your board that has loaded Android click on “Bluetooth File Transfer”



The Bluetooth File Transfer utility will be started



This utility will list all the files in your SD card. Check the file you want to send to your cell phone



Press K2 and click on "More"



Click on “Send via Bluetooth (1 file)”, a dialog will pop up and prompt you to select a target device. All Bluetooth devices will be listed including those connected or not connected. The connected devices will be checked



Click on your cell phone and click on “OK” in the following dialog



Click on “OK” you will see file transferring



You cell phone will prompt you whether or not to accept a file sent from your board. Click

on “Yes” to take it



After file transfer is done you will see the following dialog



6.1.14.3 Transfer File to Mini6410-1405

Please follow the steps described in the previous section to connect your board to a cell phone. On your board that has loaded Android click on “Bluetooth File Transfer”



The Bluetooth File Transfer utility will be started



Press K2 and click on “More”



Click on “Discoverable” and click on “Yes” to continue



Now you can send a file from your cell phone to your board. The file will be saved under

“/mnt/sdcard”



6.1.15 USB Flash Drive

Android supports plug and play of USB flash drives up to a maximum of 32G (note: the drive should be formatted to FAT32).

Insert your drive to the USB host and a flash drive icon will appear in the upper left of the screen



Pull down the task bar on the top



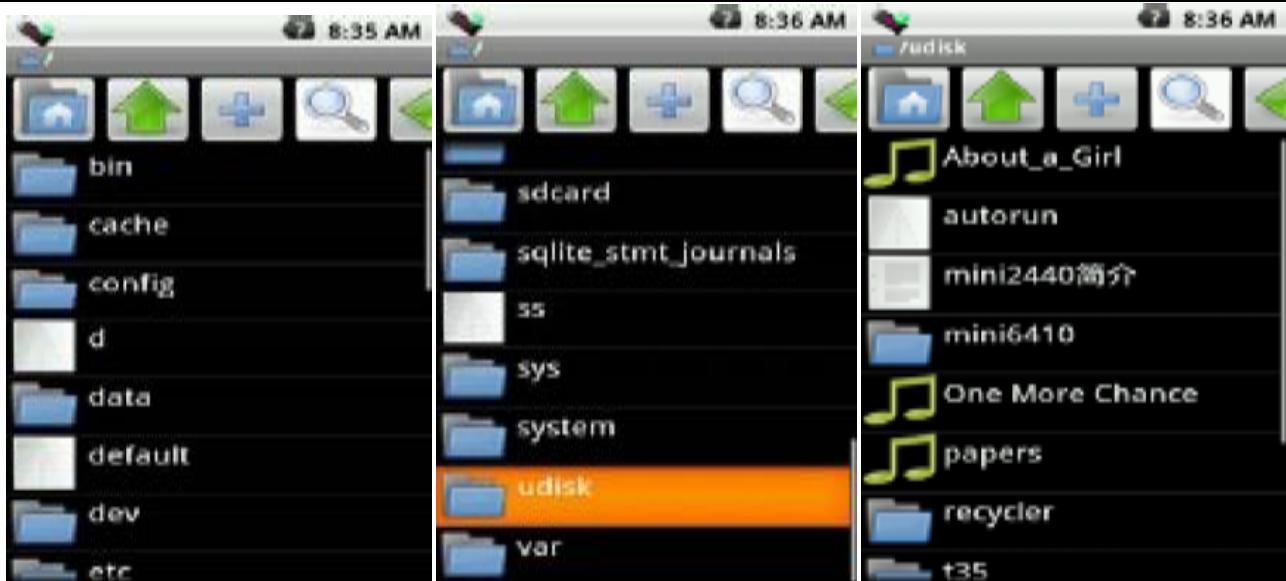
Click on the drive icon



Click on “Unmount USB mass storage” you will unmount your drive. Click on “Open folder browser” you can browse your files



Click on the green up arrow to go to the top directory. Locate “/udisk” and click to open it

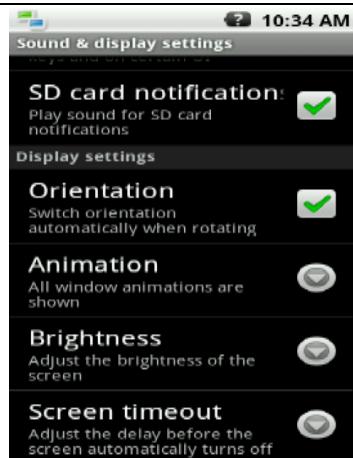


6.1.16 Backlight Control

Maybe you have noticed that after the system boots the backlight will turn off gradually if the touch screen doesn't receive any touch. This is manipulated by the backlight control.
Click on “Sound & display”



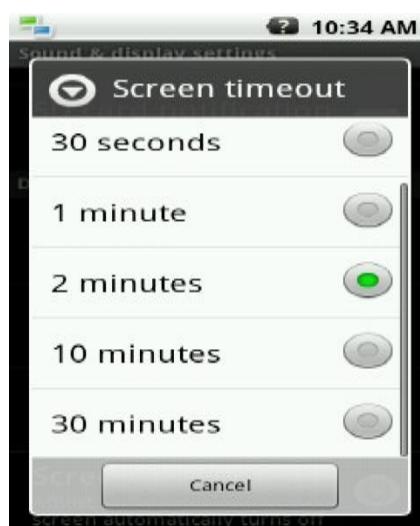
Locate “Display settings”



Click on “Brightness” you can set its brightness



Click on “Screen timeout” you can set its turn off time

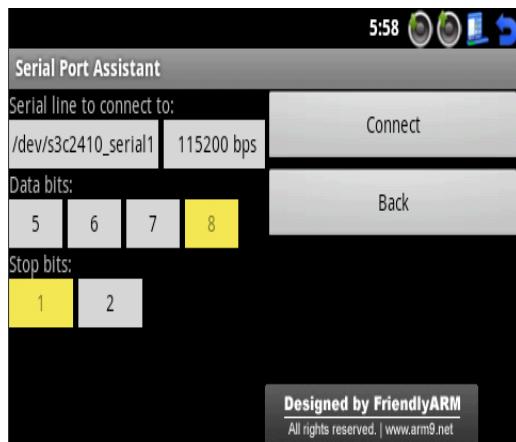


6.1.17 Serial Port Assistant

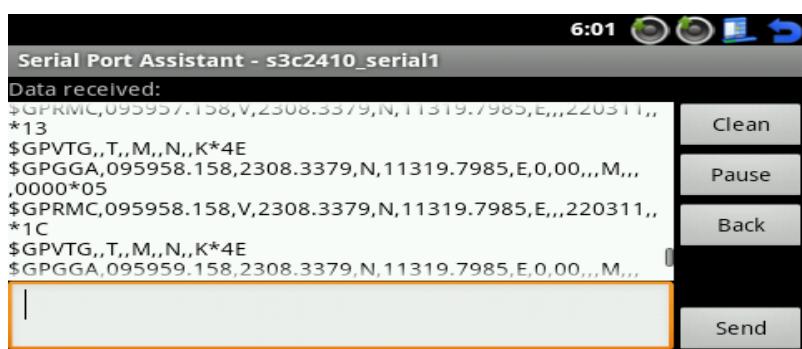
To launch our serial port assistant utility, you can click on the “iTest” icon



Click on “Serial Port Assistant” and you can set up its parameters as follows:



After setup is done, click on “Connect” and if the connection is successful you will see the following messages from the serial port



To send data to the serial port, you can type your messages in the left text box and click on “send”. “Pause” pauses messages’ popping and “Clean” removes all the received messages

6.1.18 LED Test

To test LEDs, please click on the “iTTest” icon



Click on “LED Testing” and you will see the following window and be able to test LEDs by clicking on those buttons



6.1.19 PWM Buzzer

To test PWM, please click on the iTest icon



Click on “PWM Testing” you will see the following window



On the dialog, you can type a frequency and “start” or adjust the frequency by clicking on “+” and “-”. To stop it you can click on “stop”.

6.1.20 ADC Test

To test ADC, please click on the “iTet” icon



Click on “A/D Convert” you will see the following window

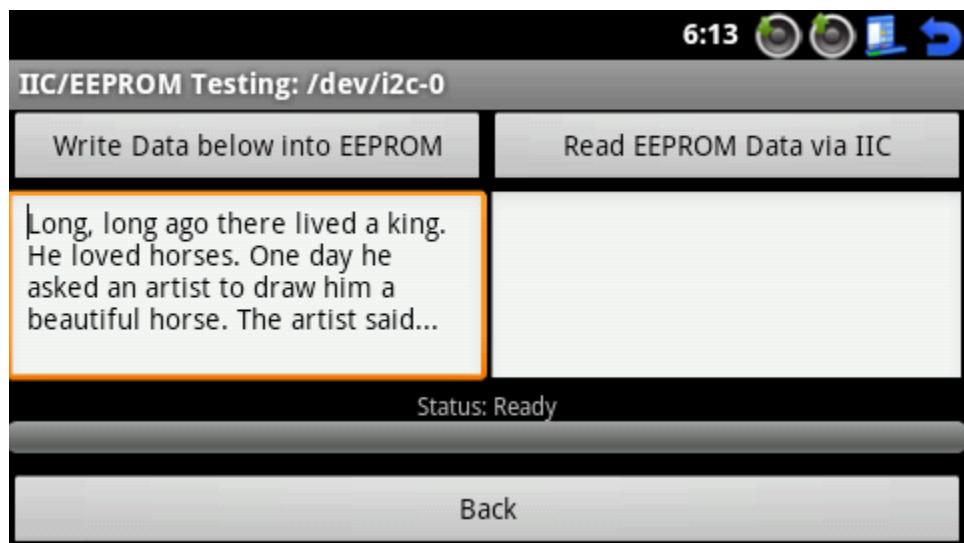


6.1.21 I2C-EEPROM

To test “I2C-EEPROM” please click on the “iTet” icon



Click on “IIC/EEPROM Testing” you will see the following window



Click on “Write Data below into EEPROM” to write your data on the left to “EEPROM”

and then click on “Read EEPROM Data via IIC” to read it from EEPROM to the right area

6.2 Set up Android Development Environment

Please refer to the materials under “development documents\02 Android programming” in the shipped CD

6.3 Set up Android Compiler

In this section we will show you how to set up the development environment and compiler for Android.

6.3.1 Android Development and Compiler

The development environment for Android is very similar to Linux. You need to install Fedora9, a cross compiler and mktools. It uses the same compiler Linux uses

Note: to compile Android your system should have at least 5G hard disk space.

6.3.2 Uncompress and Install Source Code

Let's first create a working directory “/opt/FriendlyARM/mini6410/android”

Type the command below in a terminal

```
#mkdir -p /opt/FriendlyARM/mini6410/android
```

The code that will be uncompressed in later steps will all be in this directory.

(1) Get a Copy of Android Souce Code Package

Create a temporary directory “/tmp/android” in Fedora9

```
#mkdir /tmp/android
```

Copy all the files under “Android” from the shipped CD to “/tmp/Android”

(2) Uncompress u-boot source code

Execute the following command under “/opt/FriendlyARM/mini6410/android”

```
#cd /opt/FriendlyARM/mini6410/android
```

```
#tar xvzf /tmp/android/u-boot-mini6410-20100730.tar.gz
```

This will create a “u-boot-mini6410” directory which contains a complete copy of source code

Note: 20100730 is the date when we released it

(3) Uncompress Android Kernel

Execute the command below in “/opt/FriendlyARM/mini6410/android”

```
#cd /opt/FriendlyARM/mini6410/android
```

```
#tar xvzf /tmp/android/android-kernel-2.6.36-20110215.tar.gz
```

This will create a “linux-2.6.36-android” directory which contains a complete copy of source code

Note: 20110215 is the date when we released it

(4) Uncompress Android System

Execute the command below in “/opt/FriendlyARM/mini6410/android”

```
#cd /opt/FriendlyARM/mini6410/android
```

```
#tar xvzf /tmp/android/android-2.3-fs-20110215.tar.gz
```

This will create a “Android-2.3” directory

Note: 20110215 is the date when we released it. This source code contains a copy of Android-2.3 source code and compiling scripts.

(5) Uncompress Android

Execute the command below in “/opt/FriendlyARM/mini6410/android”

```
#cd /opt/FriendlyARM/mini6410/android  
#tar xvzf /tmp/android/ rootfs_android-20110215.tar.gz
```

This will create a rootfs_android directory

Note: 20110215 is the date when we released it.

6.4 Configure and Compile U-Boot

Note: Android uses the same U-boot as Linux.

Here we take a 128M system as an example. Please follow the steps below:

Enter the U-boot source code directory and run the command below:

```
#cd /opt/FriendlyARM/mini6410/linux/u-boot-mini6410  
#make mini6410_nand_config-ram128;make
```

This will compile a U-boot.bin which supports booting from the NAND flash. To differ it from the one in the shipped CD we name it “u-boot_nand-ram128.bin”

To compile a U-boot for 256M systems please follow the steps below:

Enter the U-boot source code directory and run the command below:

```
#cd /opt/FriendlyARM/mini6410/linux/u-boot-mini6410  
#make mini6410_nand_config-ram256;make
```

This will compile a U-boot.bin which supports booting from the NAND flash. To differ it from the one in the shipped CD we name it “u-boot_nand-ram256.bin”

6.5 Configure and Compile Linux Kernel

Android's Linux kernel is a little bit different from a standard one but its configuration method is the same. If you are not familiar with configuring a Linux kernel we suggest you use our default configuration file

To compile a kernel please follow the steps below:

```
#cp config_linux_mini6410 .config
```

```
#make zImage
```

This will generate a zImage under “arch/arm/boot”.

6.6 Create Android

Compiling Android may not be an easy task for beginners. Therefore we have a complete copy of the source code and two compiling scripts: build-android and genrootfs.sh.

Execute the command below:

```
#cd /opt/FriendlyARM/mini6410/android/Android-2.3
```

```
#!/build-android
```

This will begin to compile Android-2.3. This process may take a while. We recommend users to use a multi-core CPU and Linux instead of using a simulator.

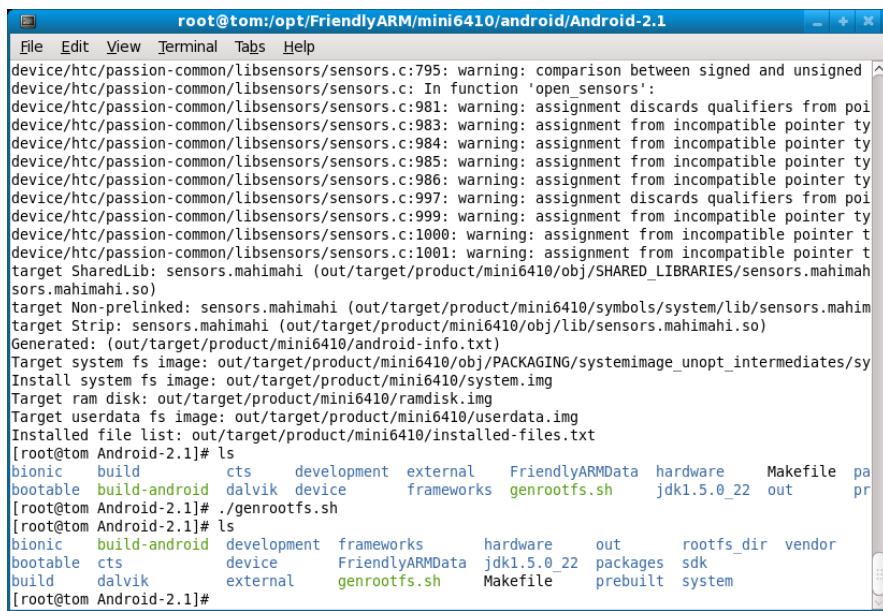
After it is done, run the following script:

```
#!/genrootfs.sh
```

This will create a target file system we need and a “rootfs_dir” directory. It is the same as

“rootfs_android”.

Note: you can compile one that supports the serial port control touch screen with the “genrootfs-s.sh” script



```
root@tom:/opt/FriendlyARM/mini6410/android/Android-2.1
File Edit View Terminal Tabs Help
device/htc/passion-common/libensors/sensors.c:795: warning: comparison between signed and unsigned
device/htc/passion-common/libensors/sensors.c: In function 'open_sensors':
device/htc/passion-common/libensors/sensors.c:981: warning: assignment discards qualifiers from poi
device/htc/passion-common/libensors/sensors.c:983: warning: assignment from incompatible pointer ty
device/htc/passion-common/libensors/sensors.c:984: warning: assignment from incompatible pointer ty
device/htc/passion-common/libensors/sensors.c:985: warning: assignment from incompatible pointer ty
device/htc/passion-common/libensors/sensors.c:997: warning: assignment discards qualifiers from poi
device/htc/passion-common/libensors/sensors.c:999: warning: assignment from incompatible pointer ty
device/htc/passion-common/libensors/sensors.c:1000: warning: assignment from incompatible pointer t
device/htc/passion-common/libensors/sensors.c:1001: warning: assignment from incompatible pointer t
target Sharedlib: sensors.mahimahi (out/target/product/mini6410/obj/SHARED_LIBRARIES/sensors.mahimah
sors.mahimahi.so)
target Non-prelinked: sensors.mahimahi (out/target/product/mini6410/symbols/system/lib/sensors.mahim
target Strip: sensors.mahimahi (out/target/product/mini6410/obj/lib/sensors.mahimahi.so)
Generated: (out/target/product/mini6410/android-info.txt)
Target system fs image: out/target/product/mini6410/obj/PACKAGING/systemimage_unopt_intermediates/sy
Install system fs image: out/target/product/mini6410/system.img
Target ram disk: out/target/product/mini6410/ramdisk.img
Target userdata fs image: out/target/product/mini6410/userdata.img
Installed file list: out/target/product/mini6410/installed-files.txt
[root@tom Android-2.1]# ls
bionic    build      cts      development  external   FriendlyARMData  hardware   Makefile  pa
bootable   build-android dalvik   device      frameworks  genrootfs.sh  jdk1.5.0_22  out      pr
[root@tom Android-2.1]# ./genrootfs.sh
[root@tom Android-2.1]# ls
bionic    build-android development frameworks  hardware   out      rootfs_dir  vendor
bootable   cts      device      FriendlyARMData jdk1.5.0_22 packages  sdk
build      dalvik   external   genrootfs.sh  Makefile  prebuilt  system
[root@tom Android-2.1]#
```

Now we have created everything we need to run Android: Bootloader, kernel and file system.

6.7 Create or Run File System

To run Android on your board you need to burn the above files into the NAND flash. The bootloader and kernel are single file images and can be burned into the flash or the SD card. The file system we just created is a directory and cannot be burned directly. Therefore we need to make it a single file image with the mktools tools.

Note: you can make an image either via an Android compiled from the source code or the one we offer in the shipped CD. The following steps are for the previous case:

6.7.1 Make YAFFS2 Image

With the **mkyaffs2image-128M** utility, you can make a yaffs2 image. The Android kernel by default supports this file system:

```
#cd /opt/FriendlyARM/mini6410/android/Android-2.3  
#mkyaffs2image-128M rootfs_dir rootfs_android.img
```

This will generate a rootfs_android.img file in the current directory.

Note: if you want to drive your serial port control touch screen you need a **rootfs_android-s image**

6.7.2 Make UBIFS Image

With the **mkubimage** utility, you can make a UBIFS image. The Android kernel by default supports this file system:

```
#cd /opt/FriendlyARM/mini6410/android/Android-2.3  
#mkubimage rootfs_dir rootfs_android.ubi
```

This will generate a rootfs_android.ubi file in the current directory.

Note: burning a UBIFS image is faster than burning a YAFFS2 image since a UBIFS image has smaller size. If you want to drive your serial port control touch screen you need a **rootfs_android-s image**

6.7.3 Make EXT3 Image

With the **mkext3image** utility, you can make an EXT3 image. You can copy it to the SD card and run it directly. The Android kernel by default supports this file system. The default FriendlyARM.ini supports this file system too:

```
#cd /opt/FriendlyARM/mini6410/android/Android-2.3
```

```
#mkext3image rootfs_dir rootfs_android.ext3
```

This will generate a rootfs_android.ext3 file. You can copy it to your SD card's "images/Android/" directory. Also you need to make sure to define "Android-RootFs-RunImage =" to this file in the FriendlyARM.ini file.

Note: the size of an EXT3 file image usually is 30% bigger than that of other images. For a file system that is less than 64M it will be treated as a 64M system. That is the minimum size of an ext3 image is $64M \times 1.3 = 83.2M$ 。

Note: If you want to drive your serial port control touch screen you need a `rootfs_android-s image`