



AXIOMTEK

rBOX610

Linux

Software User's Manual



Disclaimers

This manual has been carefully checked and believed to contain accurate information. Axiomtek Co., Ltd. assumes no responsibility for any infringements of patents or any third party's rights, and any liability arising from such use.

Axiomtek does not warrant or assume any legal liability or responsibility for the accuracy, completeness or usefulness of any information in this document. Axiomtek does not make any commitment to update the information in this manual.

Axiomtek reserves the right to change or revise this document and/or product at any time without notice.

No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Axiomtek Co., Ltd.

Trademarks Acknowledgments

Axiomtek is a trademark of Axiomtek Co., Ltd.

Windows[®] is a trademark of Microsoft Corporation.

Other brand names and trademarks are the properties and registered brands of their respective owners.

©Copyright 2014 Axiomtek Co., Ltd.

All Rights Reserved

April 2014, Version A2

Printed in Taiwan

Table of Contents

Disclaimers.....	ii
Chapter 1 Introduction.....	1
1.1 Specifications.....	2
Chapter 2 Getting Started	5
2.1 Connecting the rBOX610	5
2.1.1 Serial Console	6
2.1.2 Telnet over Ethernet	8
2.2 How to Develop a Sample Program.....	10
2.2.1 Install LTIB Toolchain	10
2.2.2 Write and Compile Sample Program.....	10
2.3 How to Put and Run a Sample Program.....	11
2.3.1 Via FTP	11
2.3.2 Via USB Flash Drive.....	12
2.3.3 Via TFTP	12
Chapter 3 The Embedded Linux	13
3.1 Embedded Linux Image Managing	13
3.1.1 System Version	13
3.1.2 System Upgrade Procedures	13
3.1.3 System Time.....	14
3.1.4 Internal RTC Time	15
3.1.5 External RTC Time	15
3.1.6 Adjusting System Time.....	15
3.2 Networking.....	16
3.2.1 FTP – File Transfer Protocol	16
3.2.2 TFTP – Trivial File Transfer Protocol.....	16
3.2.3 NFS – Network File System	16
3.2.4 WiFi (Optional)	17
Chapter 4 Programming Guide	25
4.1 librb212 API Functions	25
4.2 librb212 API Examples	33
4.2.1 Get Board ID and Power Status	33
4.2.2 COM Port Configuration	33
4.2.3 Watchdog Timer	34

4.2.4	Digital Input and Output.....	35
4.2.5	LEDs Setting	35
4.2.6	Read and Write EEPROM.....	35
4.3	CAN Bus.....	36
4.4	Compile Demo Program	38
4.4.1	Install LTIB Toolchain	38
4.4.2	Run demo program	38
Chapter 5	Board Support Package (BSP)	39
5.1	Host Development System Installation	39
5.1.1	Install Host System.....	39
5.1.2	Install LTIB.....	41
5.1.3	Compile Demo Program.....	51
5.2	U-Boot for Q7M100.....	52
5.2.1	Bootting the System with an NFS Filesystem	52
5.2.2	Bootting the System from eMMC	52
5.2.3	Reference Document	53
5.3	Additional Information	54
5.3.1	4GB eMMC Partition Layout	54
5.3.2	Compile and Build Your Program	55
Chapter 6	Application Software	57
6.1	Install the Application	57
6.2	Uninstall the Application	58

Chapter 1

Introduction

The extreme compact rBOX610 supports the low power RISC-based module (iMX287) processor with extended temperature range of -40°C to +70°C for using in wide range operating environments. Multiple built-in serial ports, high-speed LANs and USB 2.0 ports enable fast and efficient data computation, communication and acquisition. Its digital I/O feature provides users with the convenience of digital devices connection. Besides, the industrial grade IP40-rated rBOX610 meets Safety Agency requirements and has passed heavy industrial CE and FCC testing.

This user's manual is for the embedded Linux preinstalled in rBOX610. The embedded Linux is derived from Freescale BSP L2.6.35_10.12.01_SDK_source.tar.gz, which is based on Linux Kernel 2.6.35.3 and our hardware patches to suit rBOX610.

Software structure

The preinstalled embedded Linux image is located in eMMC Flash memory which is partitioned and formatted to accommodate boot loader, kernel, root filesystem and data storage. It follows standard Linux architecture to allow user to easily develop and deploy application software that follows Portable Operating System Interface (POSIX).

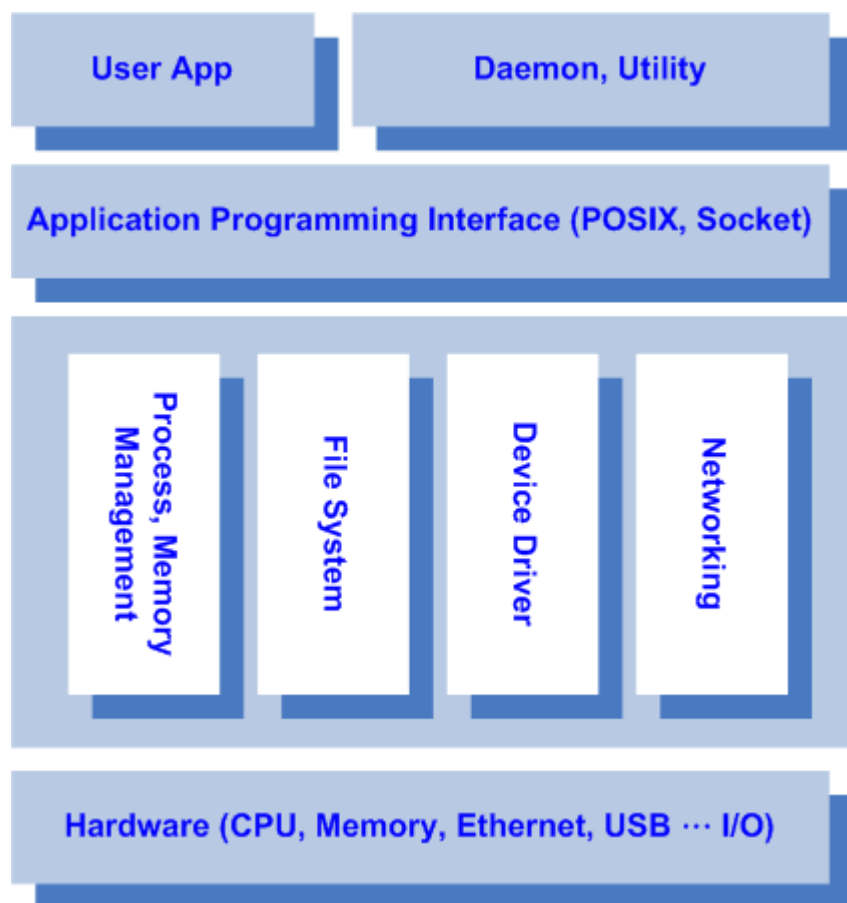
To facilitate user program in monitoring and controlling I/O device such as DIO, CAN, Watchdog Timer, the rBOX610 includes 'librb212.so' shared library.

In addition to ext2 and ext3 file system, this embedded Linux kernel is compiled with support for NFS, including server-side, client-side functionality and 'Root file system on NFS'. Using an NFS root mount we have several advantages such as:

- The root file system is not size-restricted by the device's storage like Flash memory.
- Change made to application files during development is immediately available to the target device.

For connectivity, this image includes most popular internet protocols, some servers and utilities not only making it easy for downloading/uploading files (Linux kernel, application program) or for debugging, but also communicating to outside world via Ethernet, WiFi and 3G.

For the convenience of manipulating embedded Linux, this image includes lots of popular packages such as busybox, udev, etc. Besides, we also provide Axiomtek's application software like Web App and serial server for specific use.



1.1 Specifications

- **OS: Linux**
 - Kernel: 2.6.35.3 (with Freescale and Axiomtek hardware modified patch)
- **Support Protocol Types**
 - ICMP.
 - TCP/IP.
 - UDP, DHCP, Telnet, HTTP, HTTPS, SSL, SMTP, ARP, NTP, DNS, PPP, PPPoE, FTP, TFTP, NFS.
- **Shell**
 - Busybox's ash
- **File system**
 - NFS, ext2, ext3
- **Daemons**
 - Telnetd: Telnet server daemon
 - FTPD: FTP server daemon

- **Utilities**
 - Telnet: Telnet client program
 - FTP: FTP client program
 - TFTP: Trivial File Transfer Protocol client
- **Packages**
 - **bridge-utils**: The bridge-utils package contains a utility needed to create and manage bridge devices
 - **busybox**: Small collection of standard Linux command-line utilities
 - **udev**: A device manager for Linux kernel
 - **dosfstools** : Utilities for making and checking MS-DOS FAT file system
 - **e2fsprogs**: A set of utilities for maintaining the ext2, ext3 and ext4 file systems
 - **Ethtool**: A Linux command for displaying or modifying the Network Interface Controller (NIC) parameters
 - **i2c-tools** : A heterogeneous set of I2C tools for Linux
 - **Libtool** : A computer programming tool from the GNU build system used for creating portable compiled libraries
 - **minicom**: A text-based modem control and terminal emulation program
 - **procps** : Utilities to report on the state of the system, including the states of running processes, amount of memory
 - **Timezone**: Managing time zone data
 - **wireless-tools**: A package of Linux commands (simple text-based utilities/tools) intended to support and facilitate the configuration of wireless devices using the Linux Wireless Extension
- **Development Environment**
 - Host OS/ development OS: Ubuntu 10.04 LTS
 - Toolchain/ cross compiler: ARM, gcc-4.4.4, multilib, neon optimized (Freescale LTIB)
- **Support Software Types (Optional, developed by Axiomtek)**
 - **Serial Server**
 - Support TCP Server/ TCP Client/ UDP/ Pair/ VC
 - Support IP filter
 - Support 32 TCP connections
 - Support QoS
 - **Modbus Gateway.**
 - Support Modbus TCP/ Modbus RTU/ Modbus ASCII
 - Support IP filter
 - Support 32 connections
 - Support TCP for multiple comport
 - Support QoS

- **HW's Lib (Hardware's Library)**
 - **WiFi (Optional)**
 - Detect signal strength
 - Set AP connection
 - Set web, wpa, wpa2
 - Support search AP
 - **Digital I/O**
 - Read digital input
 - Write digital output
 - **CAN**
 - Support open/write/read/close functions
 - **3G**
 - Set number connection
 - Support user name/password
 - Detect signal strength
 - **GPS**
 - Detect signal strength
 - Support satellite positioning
 - **Watch Dog Timer**
 - Enable
 - Clean
 - Set timer
 - **COM**
 - RS-232/422/485 mode setting



Note

All specifications and images are subject to change without notice.

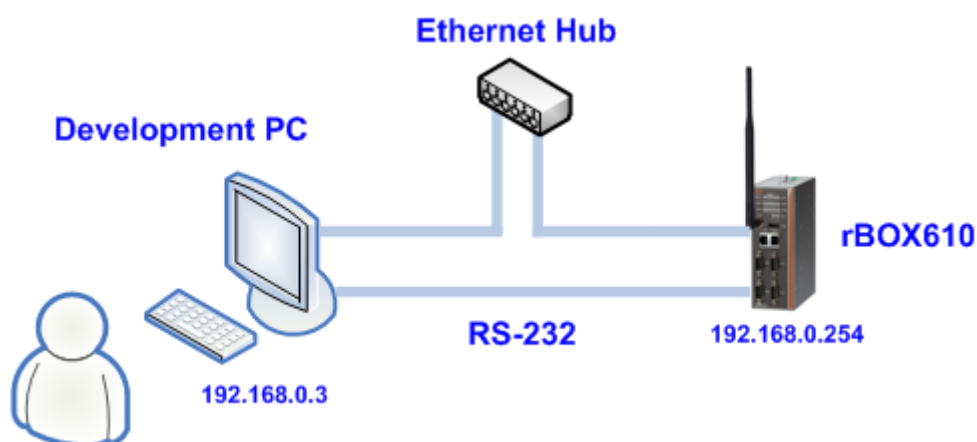
Chapter 2

Getting Started

2.1 Connecting the rBOX610

You can connect the rBOX610 to personal computer (PC) in two ways:

- Serial RS-232 console
- Telnet over Ethernet



2.1.1 Serial Console

The serial console is a convenient interface for connecting rBOX610 to PC. First of all, it is very important to make sure that the serial port settings are as follows:

Baudrate: 115200 bps

Parity: None

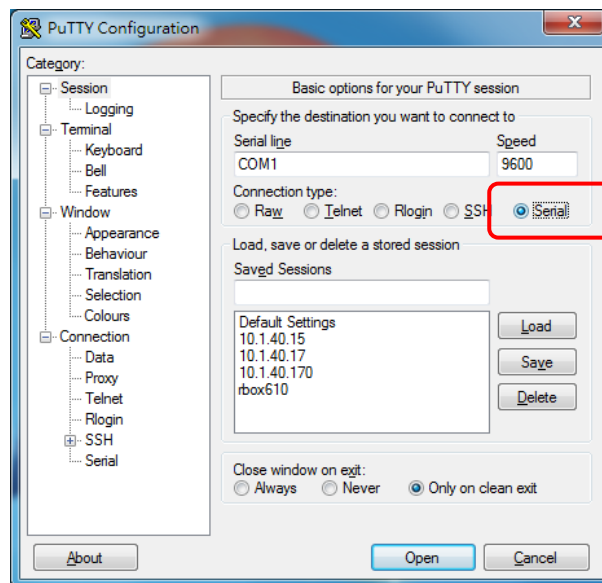
Data bits: 8

Stop bit: 1

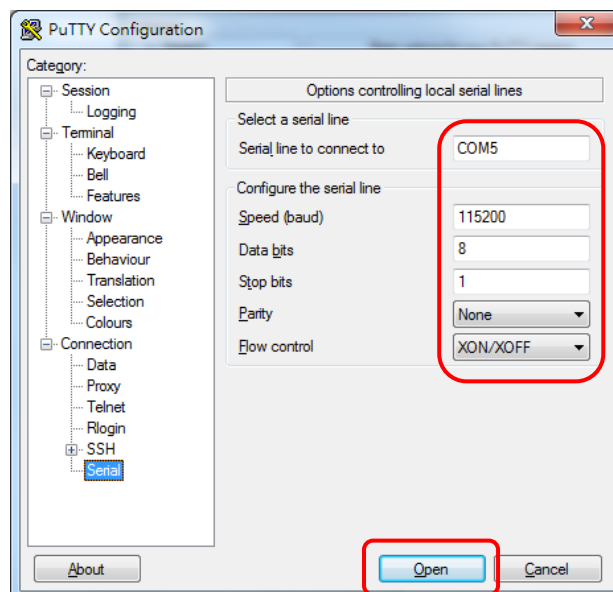
Flow Control: None

Here we use PuTTY to setup and link to the rBOX610. Learn how to do it with these step by step instructions:

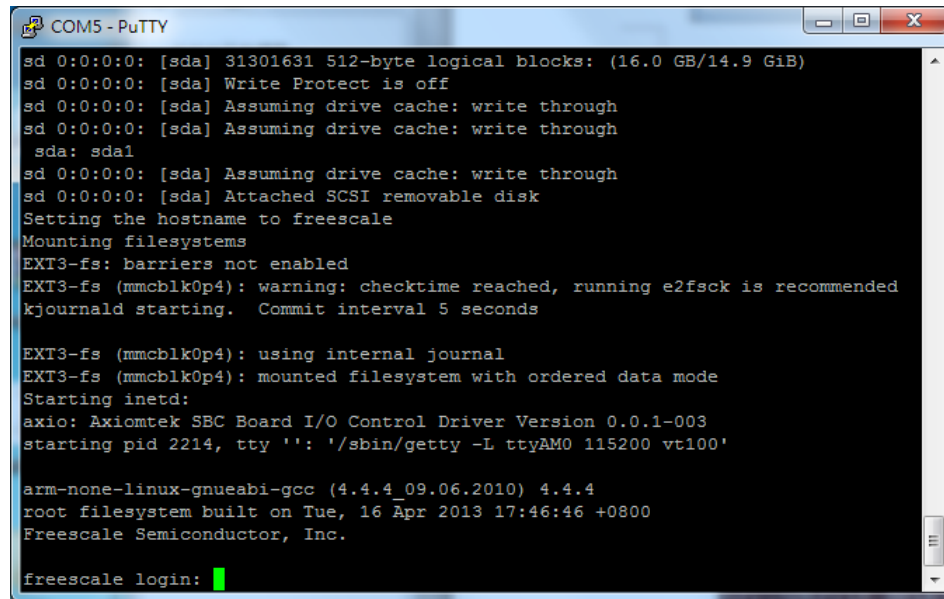
1. Open PuTTY and choose 'Serial' as the connection type.



2. Configure the serial port correctly (see image below). Click Open and power on the rBOX610.



3. If connection is established successfully, you should see the following image.



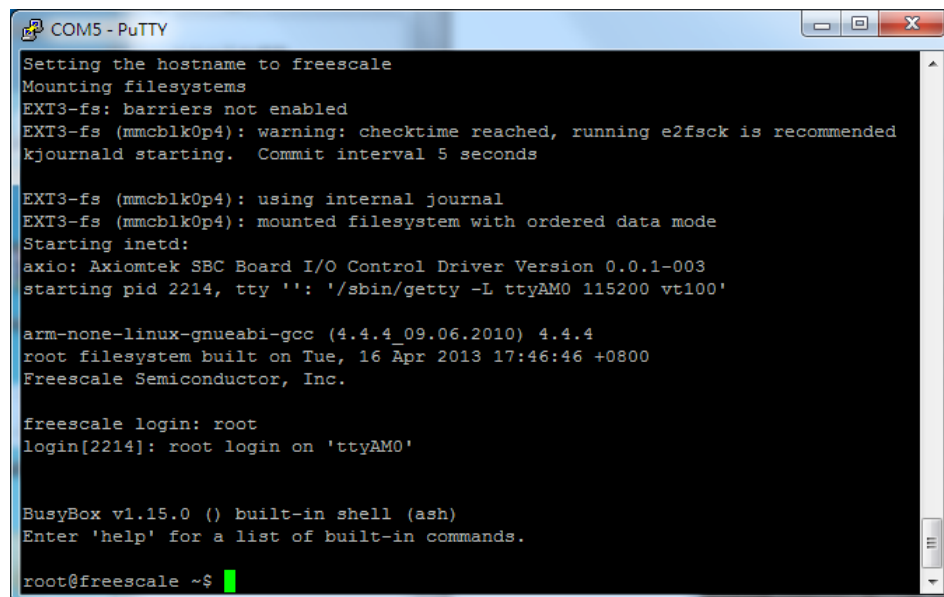
```
COM5 - PuTTY
sd 0:0:0:0: [sda] 31301631 512-byte logical blocks: (16.0 GB/14.9 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Assuming drive cache: write through
sda: sda1
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
Setting the hostname to freescale
Mounting filesystems
EXT3-fs: barriers not enabled
EXT3-fs (mmcblk0p4): warning: checktime reached, running e2fsck is recommended
kjournald starting. Commit interval 5 seconds

EXT3-fs (mmcblk0p4): using internal journal
EXT3-fs (mmcblk0p4): mounted filesystem with ordered data mode
Starting inetd:
axio: Axiomtek SBC Board I/O Control Driver Version 0.0.1-003
starting pid 2214, tty '': '/sbin/getty -L ttyAM0 115200 vt100'

arm-none-linux-gnueabi-gcc (4.4.4_09.06.2010) 4.4.4
root filesystem built on Tue, 16 Apr 2013 17:46:46 +0800
Freescale Semiconductor, Inc.

freescale login: █
```

4. To login, please enter 'root' (with no password).



```
COM5 - PuTTY
Setting the hostname to freescale
Mounting filesystems
EXT3-fs: barriers not enabled
EXT3-fs (mmcblk0p4): warning: checktime reached, running e2fsck is recommended
kjournald starting. Commit interval 5 seconds

EXT3-fs (mmcblk0p4): using internal journal
EXT3-fs (mmcblk0p4): mounted filesystem with ordered data mode
Starting inetd:
axio: Axiomtek SBC Board I/O Control Driver Version 0.0.1-003
starting pid 2214, tty '': '/sbin/getty -L ttyAM0 115200 vt100'

arm-none-linux-gnueabi-gcc (4.4.4_09.06.2010) 4.4.4
root filesystem built on Tue, 16 Apr 2013 17:46:46 +0800
Freescale Semiconductor, Inc.

freescale login: root
login[2214]: root login on 'ttyAM0'

BusyBox v1.15.0 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

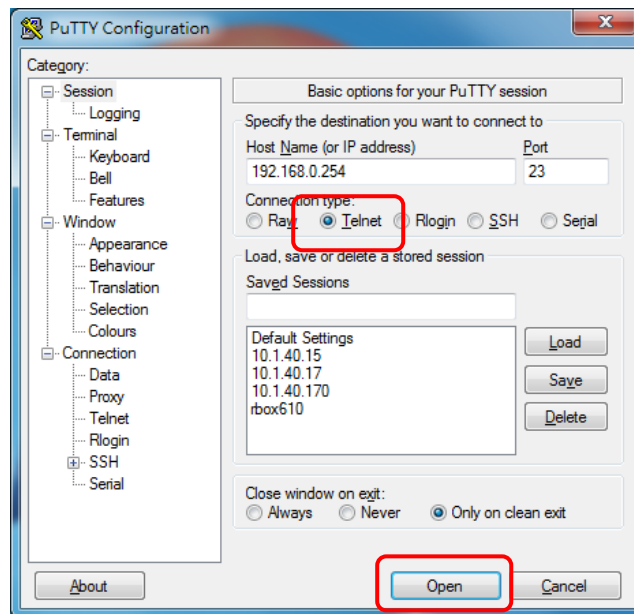
root@freescale ~$ █
```

2.1.2 Telnet over Ethernet

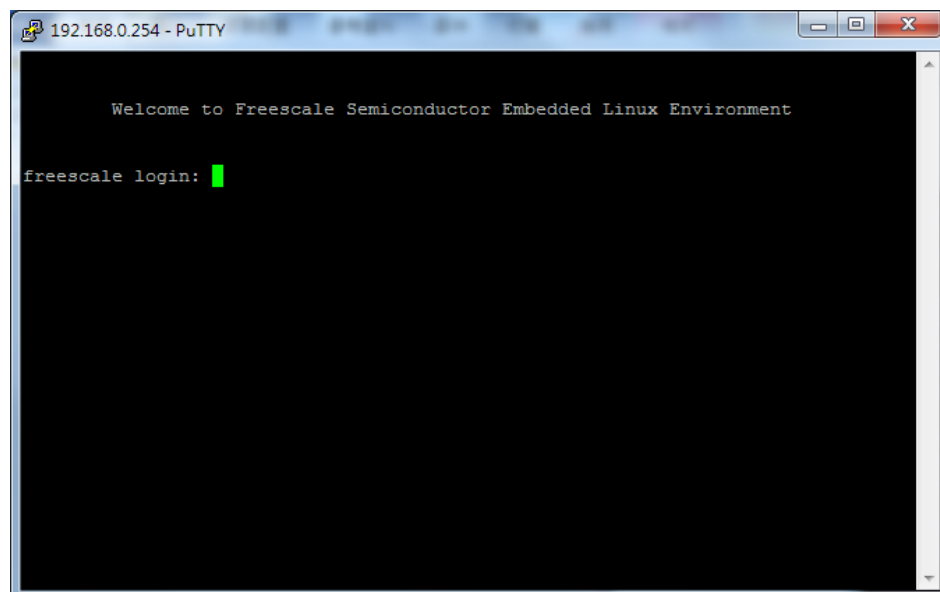
Now, we are going to connect the rBOX610 to PC over Ethernet. The following illustrations show how to do it under Windows® and Linux environment. Note that rBOX610 LAN1 default IP address is 192.168.0.254.

For Windows® users:

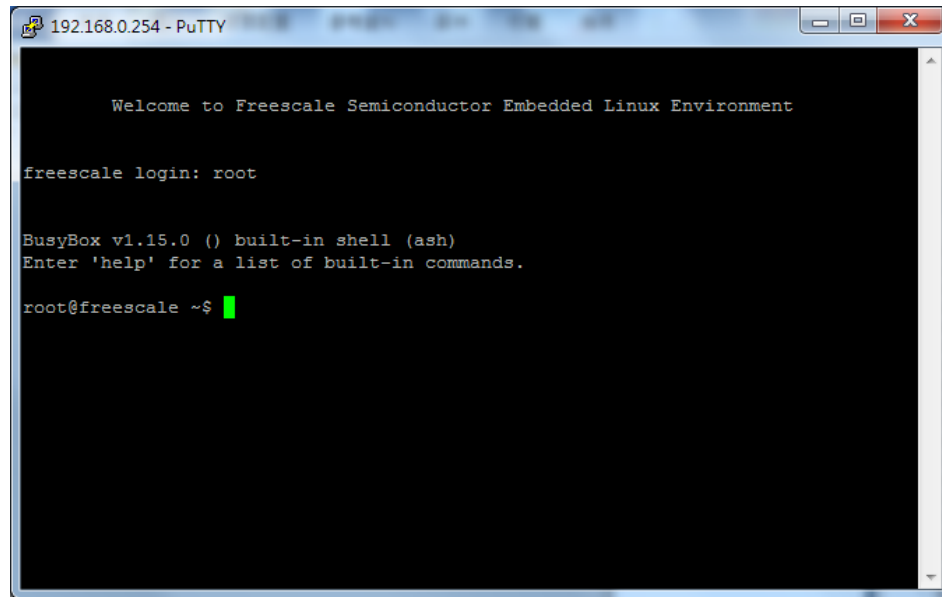
1. Here we also use PuTTY to setup and link. Open PuTTY and choose 'Telnet' as the connection type. Then set the IP address to 192.168.0.254 and click Open.



2. If connection is established successfully, you should see the following image.



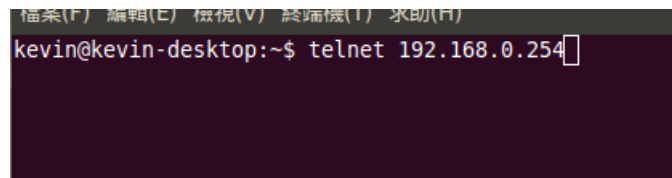
3. To login rBOX610, please enter 'root' (with no password).



A screenshot of a PuTTY terminal window titled '192.168.0.254 - PuTTY'. The terminal displays the following text: 'Welcome to Freescale Semiconductor Embedded Linux Environment', 'freescale login: root', 'BusyBox v1.15.0 () built-in shell (ash)', 'Enter 'help' for a list of built-in commands.', and 'root@freescale ~\$' with a green cursor.

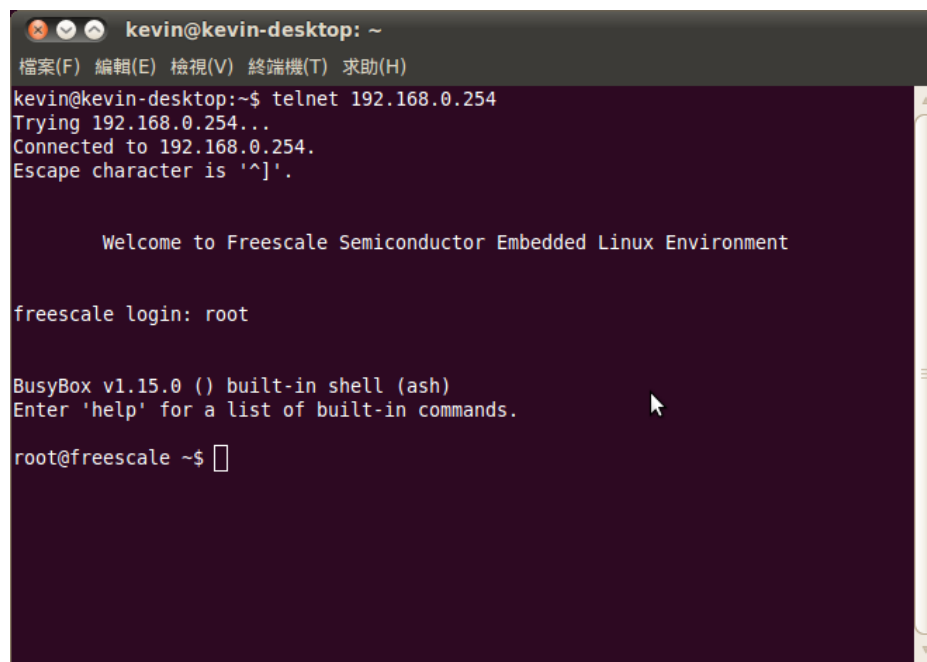
For Linux users:

1. Open terminal and keyin 'telnet' command.



A screenshot of a terminal window showing the command 'telnet 192.168.0.254' being entered at the prompt 'kevin@kevin-desktop:~\$'.

2. After the connection is established successfully, please enter 'root' to login.



A screenshot of a terminal window showing the successful login process. The terminal displays the following text: 'Welcome to Freescale Semiconductor Embedded Linux Environment', 'freescale login: root', 'BusyBox v1.15.0 () built-in shell (ash)', 'Enter 'help' for a list of built-in commands.', and 'root@freescale ~\$' with a cursor.

2.2 How to Develop a Sample Program

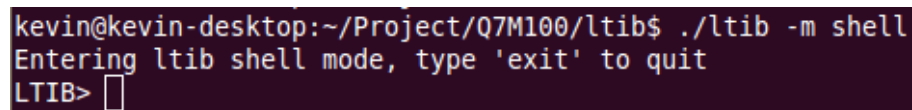
In this section, learn how to develop a sample program for rBOX610 with the following step by step instructions. The sample program is named 'hello.c'.

2.2.1 Install LTIB Toolchain

Before you develop and compile sample program, you should install Linux toolchain into development PC. To do so, refer to Chapter 5 Board Support Package.

2.2.2 Write and Compile Sample Program

```
$ cd ~/Project/Q7M100/ltib  
$./ltib -m shell
```



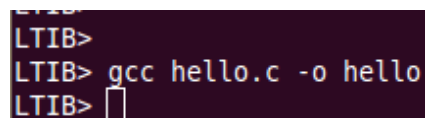
```
kevin@kevin-desktop:~/Project/Q7M100/ltib$ ./ltib -m shell  
Entering ltib shell mode, type 'exit' to quit  
LTIB> 
```

```
LTIB> cd rpm/BUILD/  
LTIB> mkdir example -p  
LTIB> cd example  
Use vi to edit hello.c.  
LTIB> vi hello.c
```

```
#include<stdio.h>  
int main()  
{  
    printf("hello world\n");  
    return 0;  
}
```

To compile the program, please do:

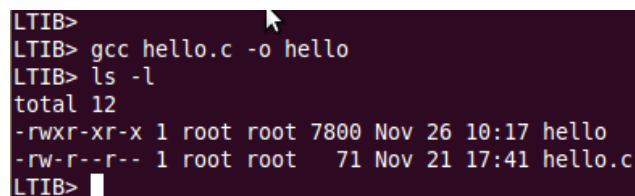
```
LTIB> gcc hello.c -o hello
```



```
LTIB>  
LTIB> gcc hello.c -o hello  
LTIB> 
```

After compiling, enter the following command and you can see the 'hello' execution file.

```
LTIB> ls -l
```



```
LTIB>  
LTIB> gcc hello.c -o hello  
LTIB> ls -l  
total 12  
-rwxr-xr-x 1 root root 7800 Nov 26 10:17 hello  
-rw-r--r-- 1 root root 71 Nov 21 17:41 hello.c  
LTIB> 
```

2.3 How to Put and Run a Sample Program

In this section, we provide 3 methods showing how to put the 'hello' program into rBOX610 and execute it.

2.3.1 Via FTP

By default, the rBOX610 comes with a built-in FTP server. Users can put 'hello' program to rBOX610 via FTP by following the steps below.

1. `$ ftp 192.168.0.254` (without username and password)

```
LTIB> ftp 192.168.0.254
Connected to 192.168.0.254.
220 Operation successful
Name (192.168.0.254:kevin):
230 Operation successful
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

2. `ftp> bin`
3. `ftp> put hello`

```
ftp> bin
200 Operation successful
ftp> put hello
local: hello remote: hello
200 Operation successful
150 Ok to send data
226 Operation successful
7800 bytes sent in 0.00 secs (35264.8 kB/s)
ftp>
```

If the operation is successful, you can see 'hello' program at rBOX610's `/root` directory.

```
root@freescale ~$ ls
hello test tools
root@freescale ~$
```

4. `$ chmod +x hello`
5. Run the 'hello' program.
`$./hello`

```
root@freescale ~$ chmod +x hello
root@freescale ~$ ./hello
hello world
root@freescale ~$
```

2.3.2 Via USB Flash Drive

Another method of putting 'hello' program into rBOX610 is via USB flash drive. Please follow the instructions below.

1. From the PC, copy 'hello' program to USB flash drive.
2. Attach USB flash drive to rBOX610.

```
$ mkdir /mnt/usb -p
$ mount /dev/sda1 /mnt/usb
```

```
root@freescale ~$
root@freescale ~$ mkdir /mnt/usb/ -p
root@freescale ~$ mount /dev/sda1 /mnt/usb/
root@freescale ~$
```

3.

```
$ cp /mnt/usb/hello /root
```
4.

```
$ cd /root
```
5.

```
$ chmod +x hello
```
6.

```
$ ./hello
```

```
root@freescale ~$
root@freescale ~$
root@freescale ~$ cp /mnt/usb/hello /root/
root@freescale ~$ udhpcp
root@freescale ~$ cd /root/
root@freescale ~$ chmod +x hello
root@freescale ~$ ./hello
hello world
root@freescale ~$
```

2.3.3 Via TFTP

Originally the Host Development System Installation already has TFTP server installed. You can put the 'hello' program into rBOX610 via TFTP. Please follow the instructions below.

1.

```
$ cp hello /tftpboot
```

```
LTIB>
LTIB> cp hello /tftpboot/
LTIB>
```

2.

```
$ tftp -g -r hello 192.168.0.3
```

 (tftp server IP)
3.

```
$ chmod +x hello
```
4. Run the 'hello' program.

```
$ ./hello
```

```
root@rBOX610 ~$
root@rBOX610 ~$ tftp -g -r hello 192.168.0.3
root@rBOX610 ~$ chmod +x hello
root@rBOX610 ~$ ./hello
hello world
root@rBOX610 ~$
```


Chapter 3

The Embedded Linux

3.1 Embedded Linux Image Managing

3.1.1 System Version

This section describes how to determine system version information including kernel and root filesystem version.

Check kernel version with the following command:

```
$ uname -r
```

```
root@freescale ~$ uname -r
2.6.35.3-Q7M100-017
root@freescale ~$
```

Check root filesystem with the following command:

```
$ grep RootFS /etc/ltib-release
```

```
root@freescale ~$ grep RootFS /etc/ltib-release
RootFS version = 1.7.1
root@freescale ~$
```

3.1.2 System Upgrade Procedures

This section describes how to upgrade kernel and root filesystem.

Upgrade kernel by following steps below:

1. Copy *ulmage* to */mnt/storage/firmware* folder:
First check whether partition is mounted on */mnt/storage*.

```
$ df -h
```

```
root@freescale ~$ df -h
Filesystem      Size      Used Available Use% Mounted on
/dev/root        1007.9M    108.8M    847.9M    11% /
tmpfs            60.6M      64.0K     60.6M     0% /dev
shm             60.6M       0        60.6M     0% /dev/shm
rwfs            512.0K       0      512.0K     0% /mnt/rwfs
/dev/mmcblk0p4   2.6G      87.6M     2.4G      3% /mnt/storage
```

Then attach the usb flash drive that contains *ulmage* file and mount it to */mnt/usb*.

```
$ mount -v /dev/sda1 /mnt/usb
```

```
root@freescale ~$ mount -v /dev/sda1 /mnt/usb
```

Copy *ulmage*.

```
$ cp /mnt/usb/ulmage /mnt/storage/firmware
```

```
root@freescale ~$ cp /mnt/usb/uImage /mnt/storage/firmware/
```

2. Run firmware update.
`$ /etc/update_fw`

```
root@freescale ~$ /etc/update_fw
Detected kernel image
Update kernel...
5284+1 records in
5285+0 records out
Update finished!
root@freescale ~$
```

Upgrade root filesystem by following steps below:

1. Copy rootfs-1.x.x.tar.gz to `/mnt/storage/firmware` folder and rename it to `rootfs.tgz`.
`$ cp /mnt/usb/rootfs-1.7.1.tar.gz /mnt/storage/firmware/rootfs.tgz`

```
root@freescale ~$ cp /mnt/usb/rootfs-1.7.1.tar.gz /mnt/storage/firmware/rootfs.tgz
```

2. Run firmware update again. After the root filesystem upgrade is complete, the system will reboot automatically
`$ /etc/update_fw`

```
root@freescale ~$ /etc/update_fw
Detected Root filesystem packages
Prepare to update system...
chroot to tmp filesystem
remove old data
```

```
Extract new data...
Remove tmp filesystem
Update finished! After 5 seconds will reboot
```



Note

You can upgrade both kernel and root filesystem at the same time by copying `ulmage` and `rootfs.tgz` to `/mnt/storage/firmware`, then run `/etc/update_fw`.

3.1.3 System Time

System time is the time value loaded from internal RTC each time the system boots up. Read system time with the following command:

`$ date`

```
root@freescale ~$ date
Thu Dec 26 05:47:46 UTC 2013
```

3.1.4 Internal RTC Time

The internal RTC time is read from iMX processor internal RTC. Note that this time value is not saved, when system power is removed.

Read internal RTC time with the following command:

```
$ rdiMXRTC
```

```
root@freescale ~$ rdiMXRTC
Getting the iMX28 RTC current date and time
iMX28 Current DateTime is 2013-12-26 5:48:24
```

3.1.5 External RTC Time

The external RTC time is read from RS5C372 external RTC. When system power is removed, this time value is kept as RS5C372 is powered by battery.

Read external RTC time with the following command:

```
root@freescale ~$ rd372RTC
RS5C372 Current DateTime is 2013-12-26 5:48:37
```

3.1.6 Adjusting System Time

Adjust system time through NTP server.

```
$ ntpclient -h time.stdtime.gov.tw -s
```

Write sync time to internal RTC

```
$ hwclock -w
```

Write internal RTC time to external RTC

```
$ wr372RTC
```

```
root@freescale ~$ ntpclient -h time.stdtime.gov.tw -s
41632 21752.118 40874.0 4.9 14726.4 34194.9 0
root@freescale ~$ hwclock -w
root@freescale ~$ wr372RTC
current iMX28 RTC time
RS5C372 Current DateTime is 2013-12-26 6:3:0
root@freescale ~$
```

3.2 Networking

3.2.1 FTP – File Transfer Protocol

FTP is a standard network protocol used to transfer files from one host to another host over TCP-based network.

The rBOX610 comes with a built-in FTP server. Section 2.1 shows the steps to put 'hello' program to rBOX610 via FTP.

3.2.2 TFTP – Trivial File Transfer Protocol

TFTP is a lightweight protocol of transfer files between a TFTP server and TFTP client over Ethernet. To support TFTP, this embedded Linux image has built-in TFTP client, so does its accompanying bootloader U-boot.

In Chapter 5, there are descriptions of TFTP server installation and kernel boot up process via TFTP. Section 2.3.3 shows you how to transfer file between server and client.

3.2.3 NFS – Network File System

NFS enables you to export a directory on an NFS server and mount that directory on remote client machine as if it were a local file system. Using NFS on target machine, we can have access to a huge number of files, libraries, and utilities during development and debugging, as well as booting up kernel.

This embedded Linux kernel is compiled with support for NFS, including server-side, client-side functionality and 'Root file system on NFS'. Section 5.1 and 5.2.1 show how to boot up embedded Linux with an NFS support.

3.2.4 WiFi (Optional)

3.2.4.1 WiFi configuration

This section contains detailed informations about WiFi configurations.

1. Boot the rBOX610 into OS with username 'root' (with no password).
2. Execute command below to load WiFi module driver and generate WiFi interface.
`$ /etc/rc.d/init.d/tdspusbcardinit start`

```

COMS - PuTTY

*****
*****          3DSP BUS Driver          *****
*****

tdspbus_init_module() enter!
3DSP USB Bus Driver   Build 16:14:14, Dec 13 2013
Version : BusU101209

tdspbus_init_module() exit,ret=0!
install wlan driver...
/etc/rc.d/init.d/tdspusbcardinit: line 39: sudo: not found
Enter [_wlan_usb_probe]
3dsp usb wlan device found:) intf 0, alt configs 1, endpoints 5
ep addr 1, type 2, in/out IN
ep addr 2, type 2, in/out OUT
ep addr 3, type 2, in/out IN
ep addr 4, type 2, in/out OUT
ep addr 9, type 3, in/out IN
tdsp_spin_lock_init is bf08a604,syscall's is bf08a604
sc is bf0e2e34
usb_info is c6d9bdd0
3D[T:WMAIN:01069][Adap_Create] dsp Initialize start...!
3D[T:WMAIN:01075][Adap_Create usbinfo is c6d9bdd0,EP_control is 0,EP_bulkout_HEAD is 4!
WLAN: Bus_new_interface: dev = c6cc1000, intf = c7f92f00;
WLAN: Bus_openInterface: enter;
register_tdspbus_usbdev be called,SerialNo = 2,version=WLU1001209!

Exit register_tdspbus_usbdev with success, SerialNo = 2!state_old_from_wb = 0, devstate = 0 ParaFlag
= 0

Leave [_wlan_usb_probe]
Enter [_wlan_usb_probe]
intf 1 is not 3dsp wlan interface
usbcore: registered new interface driver 3DSP_WLAN_USB
root@freescale ~$
  
```

`$ uwbttool -a wlan`

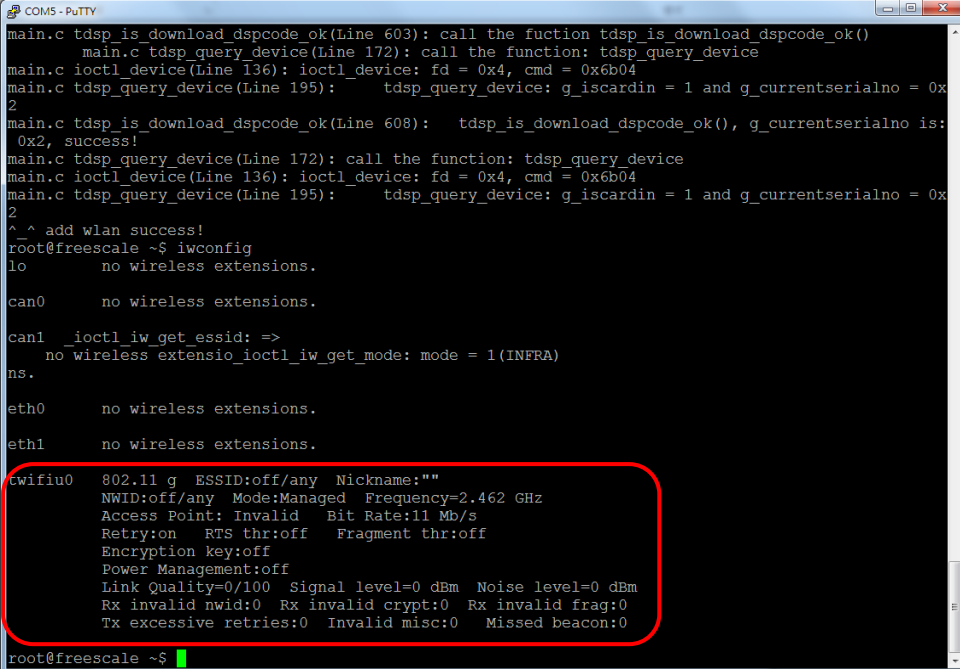
```

COMS - PuTTY

3D[T:INTRU:00411][Int_Process] :****Undefined int, type = 32, sub_type = 0,offset = 0,result = 1 **
***
3D[T:WMAIN:00825][Adap_StartDevice]: Adap GetDspFwCodesAndDownload Success!
3D[T:WMAIN:00844][Adap_StartDevice]: permanent_address=00:21:c5:1a:09:26
3D[T:WMAIN:00854][Adap_StartDevice]: current_address=00:21:c5:1a:09:26
3D[T:VENDR:02867][Vcmd_mmacSp20ResetRemoved] Resetting SP20 in Vcmd_mmacSp20ResetRemoved
3D[T:VENDR:02746][Vcmd_CardBusNICStart] start card bus nic ok
3D[T:WMAIN:05096][Adap_w2lan_init] wlan init phy mode = 2
3D[T:WMAIN:00868][Adap_StartDevice]: Adap_WLSMAC_initialize Success!
3D[T:RECEV:00224]Enter [Rx restart]
3D[T:WMAIN:00993][Adap_StartDevice]: init for usb 20 device
3D[T:WMAIN:01024][Adap_StartDevice]: Completed Init Successfully
3D[T:WMAIN:01028][Adap_StartDevice]: driver version = WLU090120_2
3D[T:WMAIN:01033][Adap_StartDevice]: 8051 version = 10.07.30.15
3D[T:WMAIN:01038][Adap_StartDevice]: dsp version = 36.02.11.104
3D[T:VENDR:02097]****SWITCHING:*****8051 go main loop *****
3D[T:INTRU:00411][Int_Process] :****Undefined int, type = db, sub_type = 50,offset = 5000,result =
190002 *****
3D[T:INTRU:00324][Int_Process] :****8051 in normal working mode****
3D[T:WMAIN:01051][Adap_StartDevice]: Vcmd_Set_Firmware_Download_Ok Successfully!
3D[T:WMAIN:01060][Adap_StartDevice]: Vcmd_Set_Encryption_Mode Successfully!
[NETDV:00235] register netdev successfully
main.c tdsp_add_device(Line 442): tdsp_add_device: success!the serialno is 2
main.c tdsp_is_download_dspscode ok(Line 603): call the fuction tdsp_is_download_dspscode_ok()
main.c tdsp_query_device(Line 172): call the function: tdsp_query_device
main.c ioctl_device(Line 136): ioctl_device: fd = 0x4, cmd = 0x6b04
main.c tdsp_query_device(Line 195): tdsp_query_device: g_iscardin = 1 and g_currentserialno = 0x
2
main.c tdsp_is_download_dspscode_ok(Line 608): tdsp_is_download_dspscode_ok(), g_currentserialno is:
0x2, success!
main.c tdsp_query_device(Line 172): call the function: tdsp_query_device
main.c ioctl_device(Line 136): ioctl_device: fd = 0x4, cmd = 0x6b04
main.c tdsp_query_device(Line 195): tdsp_query_device: g_iscardin = 1 and g_currentserialno = 0x
2
^_ add wlan success!
root@freescale ~$
  
```

```
$ iwconfig
```

If the driver loads successfully, you will see interface 'twifiu0'.



```
main.c tdsp_is_download_dspcode_ok(Line 603): call the function tdsp_is_download_dspcode_ok()
main.c tdsp_query_device(Line 172): call the function: tdsp_query_device
main.c ioctl_device(Line 136): ioctl_device: fd = 0x4, cmd = 0x6b04
main.c tdsp_query_device(Line 195):      tdsp_query_device: g_iscardin = 1 and g_currentserialno = 0x
2
main.c tdsp_is_download_dspcode_ok(Line 608):      tdsp_is_download_dspcode_ok(), g_currentserialno is:
0x2, success!
main.c tdsp_query_device(Line 172): call the function: tdsp_query_device
main.c ioctl_device(Line 136): ioctl_device: fd = 0x4, cmd = 0x6b04
main.c tdsp_query_device(Line 195):      tdsp_query_device: g_iscardin = 1 and g_currentserialno = 0x
2
^_^ add wlan success!
root@freescall ~$ iwconfig
lo          no wireless extensions.

can0        no wireless extensions.

can1        _ioctl_iw_get_essid: =>
no wireless extensio_ioctl_iw_get_mode: mode = 1(INFRA)
ns.

eth0        no wireless extensions.

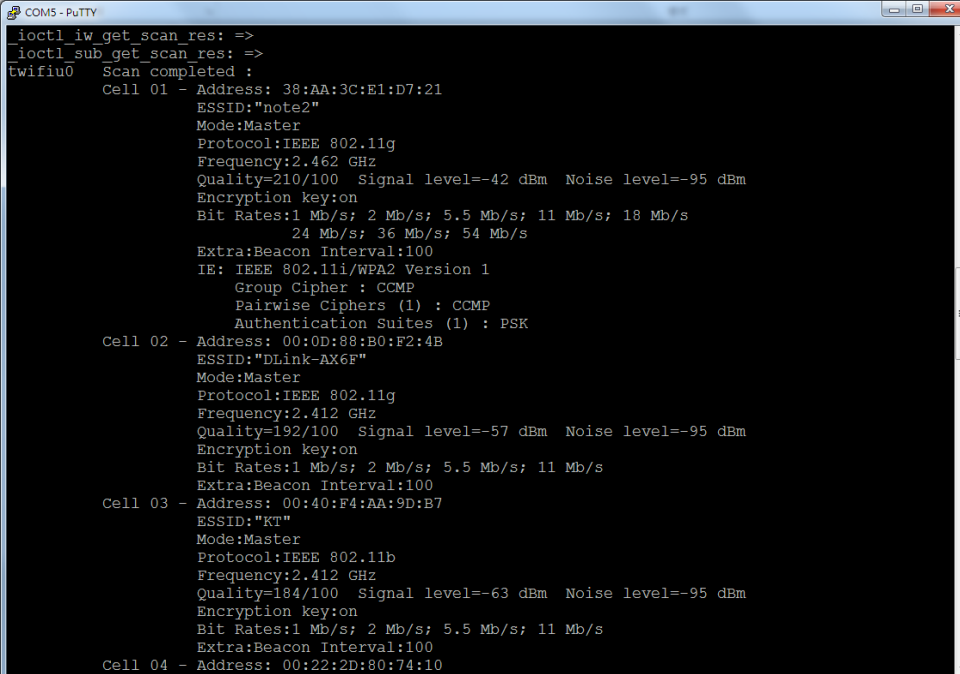
eth1        no wireless extensions.

twifiu0     802.11 g  ESSID:off/any  Nickname:""
           NWID:off/any  Mode:Managed  Frequency=2.462 GHz
           Access Point: Invalid  Bit Rate:11 Mb/s
           Retry:on  RTS thr:off  Fragment thr:off
           Encryption key:off
           Power Management:off
           Link Quality=0/100  Signal level=0 dBm  Noise level=0 dBm
           Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
           Tx excessive retries:0  Invalid misc:0  Missed beacon:0

root@freescall ~$
```

3. Scan AP.

```
$ iwlist twifiu0 scan
```



```
ioctl_iw_get_scan_res: =>
ioctl_sub_get_scan_res: =>
twifiu0     Scan completed :
Cell 01 - Address: 38:AA:3C:E1:D7:21
           ESSID:"note2"
           Mode:Master
           Protocol:IEEE 802.11g
           Frequency:2.462 GHz
           Quality=210/100  Signal level=-42 dBm  Noise level=-95 dBm
           Encryption key:on
           Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
           24 Mb/s; 36 Mb/s; 54 Mb/s
           Extra:Beacon Interval:100
           IE: IEEE 802.11i/WPA2 Version 1
               Group Cipher : CCMP
               Pairwise Ciphers (1) : CCMP
               Authentication Suites (1) : PSK
Cell 02 - Address: 00:0D:88:B0:F2:4B
           ESSID:"DLink-AX6F"
           Mode:Master
           Protocol:IEEE 802.11g
           Frequency:2.412 GHz
           Quality=192/100  Signal level=-57 dBm  Noise level=-95 dBm
           Encryption key:on
           Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
           Extra:Beacon Interval:100
Cell 03 - Address: 00:40:F4:AA:9D:B7
           ESSID:"KT"
           Mode:Master
           Protocol:IEEE 802.11b
           Frequency:2.412 GHz
           Quality=184/100  Signal level=-63 dBm  Noise level=-95 dBm
           Encryption key:on
           Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
           Extra:Beacon Interval:100
Cell 04 - Address: 00:22:2D:80:74:10
```

4. Connect AP with wpa_supplicant.

Configure wpa_supplicant:

```
$ vi example.conf
```

```
ctrl_interface=/var/run/wpa_supplicant
```

Configure encryption as disable and no authentication type.

```
network={
ssid=" AP_SSID "
key_mgmt=NONE
}
```

Configure encryption as WEP Open type.

```
network={
ssid=" AP_SSID "
key_mgmt=NONE
auth_alg=OPEN
wep_key0=0123456789
wep_tx_keyidx=0
}
```

Configure encryption as WEP and shared key as authentication type.

```
network={
ssid=" AP_SSID "
key_mgmt=NONE
auth_alg=SHARED
wep_key0=0123456789
wep_tx_keyidx=0
}
```

Configure encryption as TKIP and WPA-PSK as authentication type.

```
network={
ssid=" AP_SSID "
pairwise=TKIP
group=TKIP
proto=WPA
key_mgmt=WPA-PSK
psk="12345678"
}
```

Configure encryption as CCMP and WPA-PSK as authentication type.

```
network={
ssid=" AP_SSID "
pairwise=CCMP
group=CCMP
proto=WPA
key_mgmt=WPA-PSK
psk="12345678"
}
```

Configure encryption as CCMP and WPA2-PSK as authentication type.

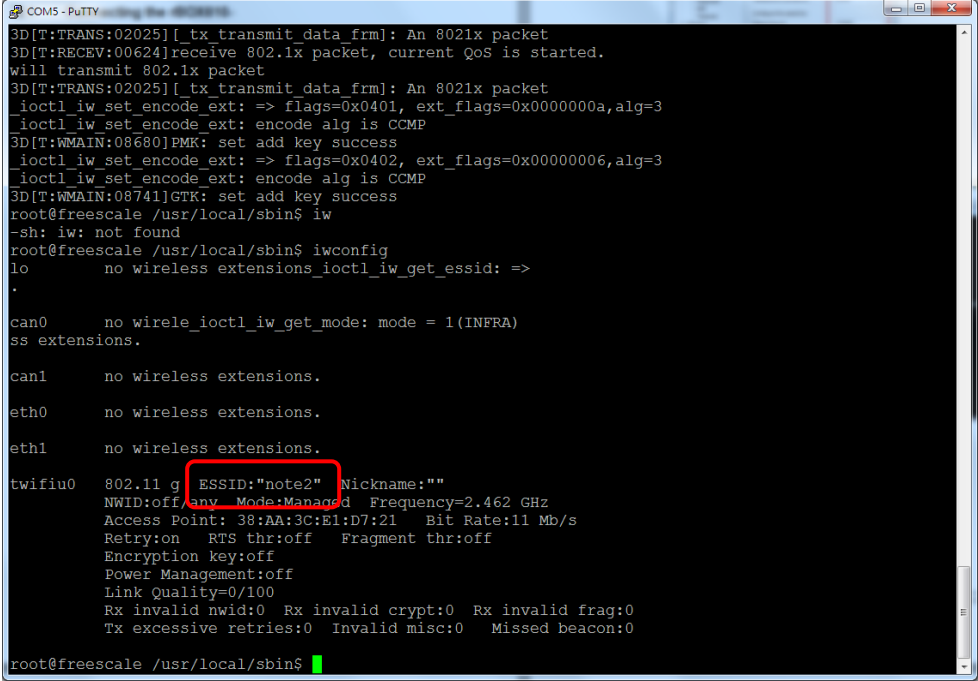
```
network={
ssid=" AP_SSID "
pairwise=CCMP
group=CCMP
proto=RSN
key_mgmt=WPA-PSK
psk="12345678"
}
```

Execute wpa_supplicant to connect to AP:

```
$ wpa_supplicant -BDwext -i twifiu0 -c example.conf
```

If connection is successful, type 'iwconfig' command and you will see image below.

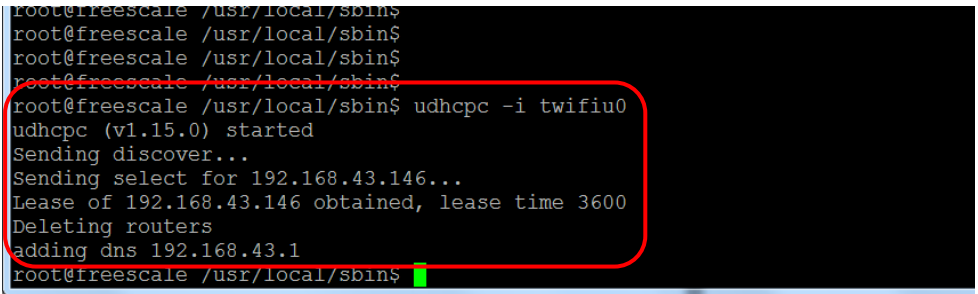
```
$ iwconfig
```



```
COMS - PuTTY
3D[T:TRANS:02025][ tx transmit data frm]: An 8021x packet
3D[T:RECEV:00624]receive 802.1x packet, current QoS is started.
will transmit 802.1x packet
3D[T:TRANS:02025][ tx transmit data frm]: An 8021x packet
ioctl iw set encode_ext: => flags=0x0401, ext_flags=0x0000000a,alg=3
ioctl iw set encode_ext: encode alg is CCMP
3D[T:WMAIN:08680]PMK: set add key success
ioctl iw set encode_ext: => flags=0x0402, ext_flags=0x00000006,alg=3
ioctl iw set encode_ext: encode alg is CCMP
3D[T:WMAIN:08741]GTK: set add key success
root@freescall /usr/local/sbin$ iw
-sh: iw: not found
root@freescall /usr/local/sbin$ iwconfig
lo        no wireless extensions_ioctl iw get_essid: =>
.
can0      no wireless extensions_ioctl iw get_mode: mode = 1(INFRA)
ss extensions.
can1      no wireless extensions.
eth0      no wireless extensions.
eth1      no wireless extensions.
twifiu0   802.11g ESSID:"note2" Nickname:""
          NWID:off,any Mode:Managed Frequency=2.462 GHz
          Access Point: 38:AA:3C:E1:D7:21 Bit Rate:11 Mb/s
          Retry:on RTS thr:off Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/100
          Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
          Tx excessive retries:0 Invalid misc:0 Missed beacon:0
root@freescall /usr/local/sbin$
```

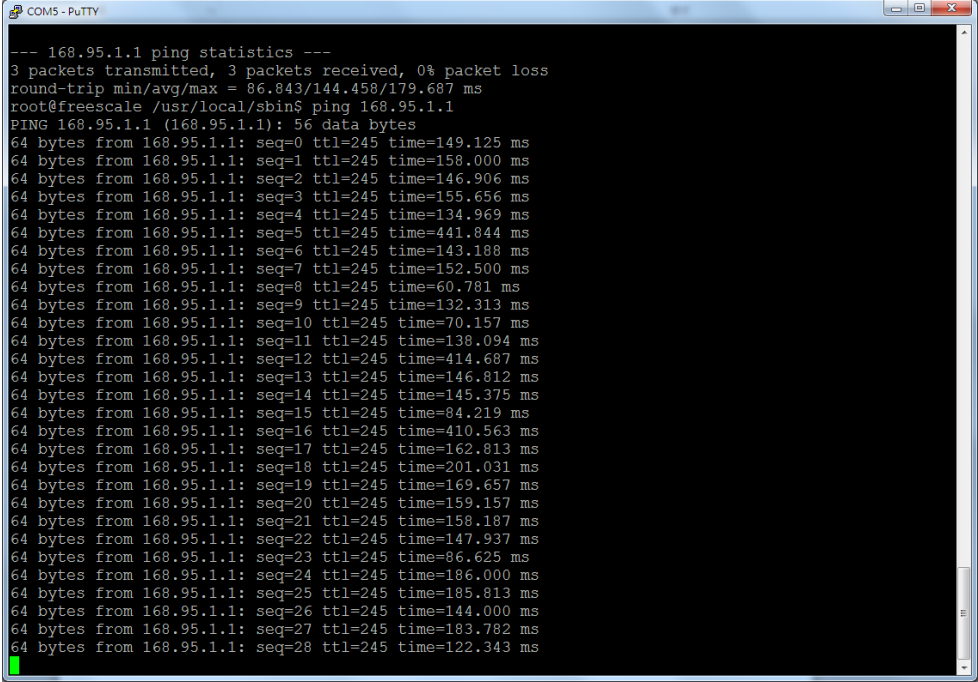
5. Get IP from AP.

```
$ udhcpc -i twifiu0
```



```
root@freescall /usr/local/sbin$
root@freescall /usr/local/sbin$
root@freescall /usr/local/sbin$
root@freescall /usr/local/sbin$ udhcpc -i twifiu0
udhcpc (v1.15.0) started
Sending discover...
Sending select for 192.168.43.146...
Lease of 192.168.43.146 obtained, lease time 3600
Deleting routers
adding dns 192.168.43.1
root@freescall /usr/local/sbin$
```


6. Test WiFi connection with ping.
\$ ping -I twifiu0 168.95.1.1



```
COM5 - PuTTY
--- 168.95.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 86.843/144.458/179.687 ms
root@freescale /usr/local/sbin$ ping 168.95.1.1
PING 168.95.1.1 (168.95.1.1): 56 data bytes
64 bytes from 168.95.1.1: seq=0 ttl=245 time=149.125 ms
64 bytes from 168.95.1.1: seq=1 ttl=245 time=158.000 ms
64 bytes from 168.95.1.1: seq=2 ttl=245 time=146.906 ms
64 bytes from 168.95.1.1: seq=3 ttl=245 time=155.656 ms
64 bytes from 168.95.1.1: seq=4 ttl=245 time=134.969 ms
64 bytes from 168.95.1.1: seq=5 ttl=245 time=441.844 ms
64 bytes from 168.95.1.1: seq=6 ttl=245 time=143.188 ms
64 bytes from 168.95.1.1: seq=7 ttl=245 time=152.500 ms
64 bytes from 168.95.1.1: seq=8 ttl=245 time=60.781 ms
64 bytes from 168.95.1.1: seq=9 ttl=245 time=132.313 ms
64 bytes from 168.95.1.1: seq=10 ttl=245 time=70.157 ms
64 bytes from 168.95.1.1: seq=11 ttl=245 time=138.094 ms
64 bytes from 168.95.1.1: seq=12 ttl=245 time=414.687 ms
64 bytes from 168.95.1.1: seq=13 ttl=245 time=146.812 ms
64 bytes from 168.95.1.1: seq=14 ttl=245 time=145.375 ms
64 bytes from 168.95.1.1: seq=15 ttl=245 time=84.219 ms
64 bytes from 168.95.1.1: seq=16 ttl=245 time=410.563 ms
64 bytes from 168.95.1.1: seq=17 ttl=245 time=162.813 ms
64 bytes from 168.95.1.1: seq=18 ttl=245 time=201.031 ms
64 bytes from 168.95.1.1: seq=19 ttl=245 time=169.657 ms
64 bytes from 168.95.1.1: seq=20 ttl=245 time=159.157 ms
64 bytes from 168.95.1.1: seq=21 ttl=245 time=158.187 ms
64 bytes from 168.95.1.1: seq=22 ttl=245 time=147.937 ms
64 bytes from 168.95.1.1: seq=23 ttl=245 time=86.625 ms
64 bytes from 168.95.1.1: seq=24 ttl=245 time=186.000 ms
64 bytes from 168.95.1.1: seq=25 ttl=245 time=185.813 ms
64 bytes from 168.95.1.1: seq=26 ttl=245 time=144.000 ms
64 bytes from 168.95.1.1: seq=27 ttl=245 time=183.782 ms
64 bytes from 168.95.1.1: seq=28 ttl=245 time=122.343 ms
```

7. Disconnect WiFi connection.
\$ wpa_cli -i twifiu0 terminate

3.2.4.2 WiFi driver compilation and installation

By default, rBox610 image will include WiFi driver. If you have to compile and install WiFi driver in newly-built image, follow the steps below:

1. download BlueW-2310U_Ubuntu12.04_3.0.10_130306.tar.gz

Note that you can get this file from Axiomtek official website:

<http://www.axiomtek.com/products/ViewDownload.asp?View=PID-rBOX610>

2. enter ltib shell mode and decompress the tar.gz file

```
$ cd ~/Project/Q7M100/ltib
```

```
$ ./ltib -m shell
```

```
LTIB> cd rpm/BUILD
```

```
LTIB> tar xzf BlueW-2310U_Ubuntu12.04_3.0.10_130306.tar.gz
```

```
LTIB> cd BlueW-2310U_3.0.10_130306
```

```
LTIB>
```

```
LTIB> ls
3dspcode.bin      README            applications      drivers
Install_3DSPusb2in1.sh  ReleaseNotes      doc              logfile
Make_3DSPusb2in1.sh  Uninstall_3DSPusb2in1.sh  driver_src      wbttool
LTIB>
```

3. compile usb bus driver

```
LTIB> cd driver_src/bus
```

```
LTIB> vi Makefile
```

Modify KDIR := (your kernel source directory)

```
else
    KDIR := /home/kevin/Project/Q7M100/kernel/linux
    PWD := $(shell pwd)
```

```
LTIB> make ARCH=arm
```

If compile success, you will see 3dspusbbus.ko in the directory

```
LTIB> ls
3dspusbbus.ko      3dspusbbus.o
3dspusbbus.mod.c   CVS
3dspusbbus.mod.o   ChangeLog.txt
LTIB>
```

4. compile wlan driver

```
LTIB> cd driver_src/wlan
```

```
LTIB> vi Makefile
```

Modify KDIR := (your kernel source directory)

```
PWD := $(shell pwd)
KDIR := /home/kevin/Project/Q7M100/kernel/linux
all:
    cp -f ../bus/Module.symvers .
    $(MAKE) -C $(KDIR) M=$(PWD) modules
```

```
LTIB> make ARCH=arm
```

If compile success, you will see 3dspusbwlan.ko in the directory

```
LTIB> ls
3dspusbwlan.ko      tdsp_bus.c
3dspusbwlan.mod.c   tdsp_bus.h
3dspusbwlan.mod.o   tdsp_bus.o
3dspusbwlan.o        tdsp_debug.c
CVS                  tdsp_debug.h
```

5. compile uwbttool and modify wbusb.conf

```
LTIB> cd applications/uwbttool/
```

```
LTIB> make
```

If compile you will see the binary file uwbttool in the directory

```
LTIB> ls
3dspusbWbtool.desktop  ChangeLog.txt  callbacks.h  main.o  uwbhotkey  wbusb.conf
BuffPrint.h           Makefile      callbacks.o  myhead.h  uwbttool
CVS                   callbacks.c   main.c      readme   uwbttool.1.gz
LTIB> █
```

```
LTIB> vi wbusb.conf
```

Change ALLOWEDMODES=3 to ALLOWEDMODES=2

```
DEFAULTMODE=3
ALLOWEDMODES=2█
SHOWWARNING=0
COEXISTWARNING=The Coexist mode is only for testing!;Ma
; max = 50 chars
```

6. edit a script file to mount driver

```
$ vi tdspusbcardinit
```

```
#!/bin/sh
```

```
#
```

```
# chkconfig: 2345 98 89
```

```
# description: 3DSP usb card initialization
```

```
#
```

```
#
```

```
### BEGIN INIT INFO
```

```
# Provides: tdspusbcardinit
```

```
# Required-Start:    $bluetooth
```

```
# Required-Stop:
```

```
# Default-Start:    2 3 4 5
```

```
# Default-Stop:     0 1 6
```

```
# Short-Description: 3DSP usb card initialization
```

```
### END INIT INFO
```

```
#set -e
```

```
case "$1" in
```

```
start)
```

```
modprobe 3dspusbbus
```

```
echo "install wlan driver...";
```

```
modprobe 3dspusbwlan
```

```
sleep 1
```

```
mknod /dev/tdspusbbus c `awk '$2=="3dspusbbus" {print $1}' /proc/devices`
```

```
0
```

```
;;
```

```
stop)
```

```
rm -f /dev/tdspusbbus
```

```
rmmod 3dspusbwlan
```

```
rmmod 3dspusbbus
```

```
;;
```

```
restart|force-reload)
```

```
$0 stop
```

```
$0 start
```

```
;;
```

```
*)
```

```
exit 1
```

```
;;
```

```
esac
```

```
exit 0
```

7. copy 3dspcode.bin, uwbttool, 3dspusbbus.ko, 3dspusbwlan.ko, wbusb.conf, tdspusbcardininit to rBOX610 via usb flash drive or ftp


```
$ mkdir -p /lib/modules/2.6.35.3-Q7M100-017/kernel/drivers/wifi/3DSP
$ cp *.ko /lib/modules/2.6.35.3-Q7M100-017/kernel/drivers/wifi/3DSP
$ mkdir -p /usr/local/3DSP/usb
$ cp wbusb.conf /usr/local/3DSP/usb/
$ cp 3dspcode.bin /usr/local/3DSP/usb
$ cp uwbttool /usr/local/sbin
$ cp tdspusbcardininit /etc/rc.d/init.d
$ chmod +x /etc/rc.d/init.d/tdspusbcardininit
```
8. modify module.dep in rBOX610


```
$ vi /lib/modules/2.6.35.3-Q7M100-017/modules.dep
```

 Add following item
 /lib/modules/2.6.35.3-Q7M100-017/kernel/drivers/wifi/3DSP/3dspusbwlan.ko:
 /lib/modules/2.6.35.3-Q7M100-017/kernel/drivers/wifi/3DSP/3dspusbbus.ko
 /lib/modules/2.6.35.3-Q7M100-017/kernel/drivers/wifi/3DSP/3dspusbbus.ko:
9. Excute tdspusbcardininit and tool uwbttool to initial wifi module


```
$ /etc/rc.d/init.d/tdspusbcardininit start
$ uwbttool -a wlan
```

```
COM5 - PuTTY
3D[T:WMAIN:00346]Download DSP: Port 20 size 0x508 mini Dsp code
3D[T:WMAIN:00825][Adap_StartDevice]: Adap_GetDspFwCodesAndDownload Success!
3D[T:WMAIN:00844][Adap_StartDevice]: permanent_address=00:21:c5:1a:08:59
3D[T:WMAIN:00854][Adap_StartDevice]: current_address=00:21:c5:1a:08:59
3D[T:VENDR:02867][Vcmd_mmacSp20ResetRemoved] Resetting SP20 in Vcmd_mmacSp20ResetRemoved
3D[T:VENDR:02746][Vcmd_CardBusNICStart] start card bus nic ok
3D[T:WMAIN:05096][Adap_w2lan_init] wlan_init phy mode = 2
3D[T:WMAIN:00868][Adap_StartDevice]: Adap_WLSMAC_initialize Success!
3D[T:RECEV:00224]Enter [Rx_restart]
3D[T:WMAIN:00993][Adap_StartDevice]: init for usb 20 device
3D[T:WMAIN:01024][Adap_StartDevice]: Completed Init Successfully
3D[T:WMAIN:01028][Adap_StartDevice]: driver version = WL090120 2
3D[T:WMAIN:01033][Adap_StartDevice]: 8051 version = 10.07.30.15
3D[T:WMAIN:01038][Adap_StartDevice]: dsp version = 36.04.11.084
3D[T:VENDR:02097]****SWITCHING:*****8051 go main loop *****
3D[T:INTRU:00411][Int_Process] :*****Undefined int, type = db, sub_type = 50,offset = 5000,
result = 190002 *****
3D[T:INTRU:00324][Int_Process] :*****8051 in normal working mode****
3D[T:WMAIN:01051][Adap_StartDevice]: Vcmd_Set_Firmware_Download_Ok Successfully!
3D[T:WMAIN:01060][Adap_StartDevice]: Vcmd_Set_Encryption_Mode Successfully!
[NETDV:00235] register netdev successfully
main.c tdsp_add_device(Line 442): tdsp_add device: success!the serialno is 2
main.c tdsp_is_download_dspcode_ok(Line 603): call the fuction tdsp_is_download_dspcode_ok(
)
main.c tdsp_query_device(Line 172): call the function: tdsp_query_device
main.c ioctl_device(Line 136): ioctl_device: fd = 0x4, cmd = 0x6b04
main.c tdsp_query_device(Line 195): tdsp_query_device: g_iscardin = 1 and g_currentseri
alno = 0x2
main.c tdsp_is_download_dspcode_ok(Line 608): tdsp_is_download_dspcode_ok(), g_currentser
ialno is: 0x2, success!
main.c tdsp_query device(Line 172): call the function: tdsp_query_device
main.c ioctl_device(Line 136): ioctl_device: fd = 0x4, cmd = 0x6b04
main.c tdsp_query_device(Line 195): tdsp_query_device: g_iscardin = 1 and g_currentseri
alno = 0x2
^^ add wlan success!
root@freescale ~$
```

Chapter 4

Programming Guide

We release a set of application programming interface (API) functions for users to access/control hardware. With these API functions, users can more easily design their own software. This chapter includes detailed description of each API function and step-by-step code samples showing how it works.

4.1 librb212 API Functions

The rBOX610 BSP includes 'librb212.so' shared library for users to access I/O and read back system information. This shared library is kept in BSP, you can find it in rBOX610-rb_lib-1.2.0.tar.bz2 of AxTools. Extract the compressed file, then besides the shared library you can also see a *demo* folder containing API header file and example programs.

Summary table of available API functions

No.	Function	Description
1	Get_BoardID()	Get board ID.
2	Get_PowerStatus()	Get power status.
3	Set_LEDStatus()	Set system LED status.
4	Get_COMType()	Get COM port communication mode type.
5	Set_COMType()	Set COM port communication mode type.
6	Set_COMTermination()	Set termination of specified COM port.
7	CPLD_Get_WDTCOUNTER()	Get watchdog timer counter value.
8	CPLD_WDT_Enable()	Enable watchdog timer. Also use it to reset WDT counter.
9	CPLD_WDT_Disable()	Disable watchdog timer.
10	CPLD_WDTStatus()	Detect watchdog timer status.
11	Get_DI()	Read high or low state on digital input channels.
12	Get_DO()	Read high or low state on digital output channels.
13	Set_DO()	Set digital output channels to high or low state.
14	Set_3GSLEDGreen()	Set 3G signal strength LED to green.
15	Set_3GSLEDRed()	Set 3G signal strength LED to red.
16	Set_3GSLEDOff()	Set 3G signal strength LED off.
17	Set_3GLLED()	Set 3G linked LED on or off.
18	Read_EEPROM_Byte()	Read byte data from EEPROM.
19	Write_EEPROM_Byte()	Write byte data to EEPROM.

Function: Get_BoardID()

Function	<code>__u16 Get_BoardID(void);</code>
Description	Get board ID.
Arguments	None.
Return	Board ID (in 2 bytes pattern).
Others	None.

Function: Get_PowerStatus()

Function	<code>int Get_PowerStatus(int number);</code>
Description	Get power status.
Arguments	number: Power number. 0: Power1. 1: Power2.
Return	0: Power fails. 1: Power is OK.
Others	None.

Function: Set_LEDStatus()

Function	<code>int Set_LEDStatus(int onoff);</code>
Description	Set system LED status.
Arguments	onoff: System LED status. 1: On. 2: Off.
Return	0: No error. 1: Function fails.
Others	None.

Function: Get_COMType()

Function	int Get_COMType(int number);
Description	Get COM port communication mode type.
Arguments	number: COM port number. 1: COM1. 2: COM2. 3: COM3. 4: COM4.
Return	0: Reserved. 1: RS232 Enable. 2: RS422/RS485_4W Enable. 3: RS485_2W Enable.
Others	None.

Function: Set_COMType()

Function	int Set_COMType(int number, int type);
Description	Set COM port communication mode type.
Arguments	number: COM port number. 1: COM1. 2: COM2. 3: COM3. 4: COM4. type: COM port mode type 0: Reserved. 1: RS232 Enable. 2: RS422/RS485_4W Enable. 3: RS485_2W Enable.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_COMTermination()

Function	int Set_COMTermination(int number, int onoff);
Description	Set termination of specified COM port.
Arguments	number: COM port number. 1: COM1. 2: COM2. 3: COM3. 4: COM4. onoff: Enable or disable termination. 1: Enable termination. 0: Disable termination.
Return	0: No error. 1: Function fails.
Others	None.

Function: CPLD_Get_WDTCOUNTER()

Function	<code>__u8 CPLD_Get_WDTCOUNTER(void);</code>
Description	Get watchdog timer counter value.
Arguments	None
Return	0~255 where 1 unit ~= 250ms.
Others	None.

Function: CPLD_WDT_Enable()

Function	<code>int CPLD_WDT_Enable(__u8 timeout);</code>
Description	Enable watchdog timer. Also use it to reset WDT counter.
Arguments	timeout: Timeout value. The range is from 0 to 255 where 1 unit ~= 250ms.
Return	0: No error. 1: Function fails.
Others	None.

Function: CPLD_WDT_Disable()

Function	<code>int CPLD_WDT_Disable(void);</code>
Description	Disable watchdog timer.
Arguments	None.
Return	0: No error. 1: Function fails.
Others	None.

Function: CPLD_WDTStatus()

Function	<code>__u8 CPLD_WDTStatus(void);</code>
Description	Detect watchdog timer status.
Arguments	None.
Return	0: If watchdog timer has not been triggered, return value = 0. 1: If watchdog timer has been triggered, return value = 1.
Others	None.

Function: Get_DI()

Function	int Get_DI(__u8 *data);
Description	Read high or low state on digital input channels.
Arguments	data: This function will store digital input data in this argument.
Return	0: No error. 1: Function fails.
Others	None.

Function: Get_DO()

Function	int Get_DO(__u8 *data);
Description	Read high or low state on digital output channels.
Arguments	data: This function will store digital output data in this argument.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_DO()

Function	int Set_DO(__u8 data);
Description	Set digital output channels to high or low state.
Arguments	data: Data to be written to digital output channels.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_3GSLEDDGreen()

Function	int Set_3GSLEDDGreen(void);
Description	Set 3G signal strength LED to green.
Arguments	None.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_3GSLEDDRed()

Function	int Set_3GSLEDDRed(void);
Description	Set 3G signal strength LED to red.
Arguments	None.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_3GSLEDDOff()

Function	int Set_3GSLEDDOff(void);
Description	Set 3G signal strength LED off.
Arguments	None.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_3GLLED()

Function	int Set_3GLLED(int onoff);
Description	Set 3G linked LED on or off.
Arguments	onoff: 3G linked LED state. 1: On. 0: Off.
Return	0: No error. 1: Function fails.
Others	None.

Function: Read_EEPROM_Byte()

Function	<code>__u8 Read_EEPROM_Byte(int address);</code>
Description	Read byte data from EEPROM.
Arguments	address: Address of data.
Return	Data read back from EEPROM.
Others	None.

Function: Write_EEPROM_Byte()

Function	<code>int Write_EEPROM_Byte(int address, __u8 value);</code>
Description	Write byte data to EEPROM.
Arguments	address: Address of data. value: data value to be written into EEPROM.
Return	0: No error. 1: Function fails.
Others	None.

4.2 librb212 API Examples



Note

Before using the API functions, remember to include header file librb212.h.

4.2.1 Get Board ID and Power Status

Use these system API functions to read boardID, check power status, and etc.

```
printf("This Board Device ID: %x\n\n", Get_BoardID());
printf("Check Power 1 status: ");
if(Get_PowerStatus(1) == 1) printf("Good!!\n\n");
else printf("Fail!!\n\n");

printf("Check Power 2 status: ");
if(Get_PowerStatus(2) == 1) printf("Good!!\n\n");
else printf("Fail!!\n\n");
```

4.2.2 COM Port Configuration

The COM port related API functions enable users to configure specified COM port communication mode to RS-232, 4-wired RS-422/485 and 2-wired RS-485.

```
printf("Read COM1 type: ");
if(Get_COMType(1) == 1) printf(" RS232\n\n");
else if(Get_COMType(1) == 2) printf(" RS422\n\n");
else if(Get_COMType(1) == 3) printf(" RS485\n\n");
else printf(" RSVD\n\n");

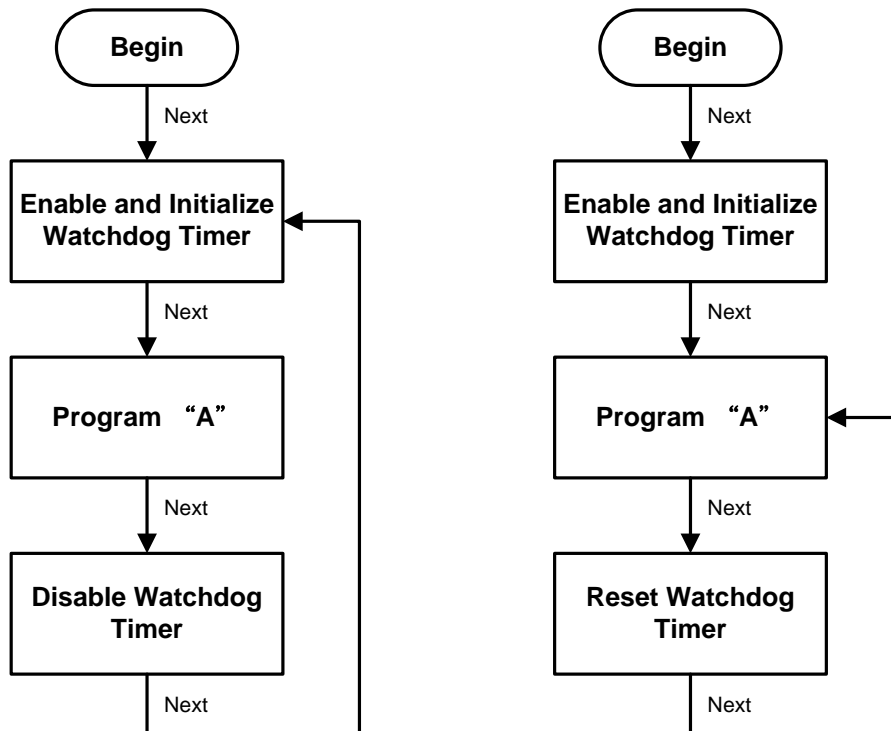
printf("Set COM2 to RS485..\n");

Set_COMType(2, 3);
printf("Read COM2 type: ");
if(Get_COMType(2) == 1) printf(" RS232\n\n");
else if(Get_COMType(2) == 2) printf(" RS422\n\n");
else if(Get_COMType(2) == 3) printf(" RS485\n\n");
else printf(" RSVD\n\n");
```

4.2.3 Watchdog Timer

Software stability is major issue in most application. Some embedded systems are not watched by human for 24 hours. It is usually too slow to wait for someone to reboot when system hangs. The systems need to be able to reset automatically when things go wrong. The watchdog timer gives us solution.

The watchdog timer is a counter that triggers a system reset when it counts down to zero from a preset value. The software starts counter with an initial value and must reset it periodically. If the counter ever reaches zero which means the software has crashed, the system will reboot. With these API functions, you can enable, disable, set watchdog timer value to 0 from 255 where 1 unit \approx 250ms, and etc..



```

printf("Enable CPLD WDT 30 sec..\n");
CPLD_WDT_Enable(120);
printf("CPLD WDT Counter: [%.2f] sec\n\n", (float)
CPLD_Get_WDTCOUNTER()/4);
printf("Delay 5 sec..\n");
sleep(5);

printf("CPLD WDT Counter: [%.2f] sec\n\n", (float)
CPLD_Get_WDTCOUNTER()/4);
printf("Disable CPLD WDT..\n");
CPLD_WDT_Disable();
  
```

4.2.4 Digital Input and Output

The digital related API functions allow users to set digital output signals to high or low and detect signal state from each digital channel.

```
unsigned char xch;
Get_DI(&xch);
printf("Current Digital-Input Data is %2X\n",xch);
Set_DO(0x3);
Get_DO(&xch);
printf("Current Digital-Output Data is %2X\n",xch);
```

4.2.5 LEDs Setting

Three different LEDs are supported by rBOX610: system LED, 3G signal strength LED and 3G linked LED. Use the LED API functions to set LED on/off state and color.

```
printf("Turn on sys LED ..\n");
Set_LEDStatus(1);
sleep(2);
printf("Turn off sys LED ..\n");
Set_LEDStatus(0);
printf("Turn on 3G Linked LED and 3G Strength LED Green..\n");
Set_3GLLED(1);
Set_3GSLEDDGreen();
sleep(2);
printf("Turn on 3G Strength LED Red..\n");
Set_3GSLEDDRed();
sleep(2);

printf("Turn off 3G Linked LED and 3G Strength LED..\n");
Set_3GLLED(0);
Set_3GSLEDDoff();
```

4.2.6 Read and Write EEPROM

The rBOX610 supports capability to store data that must be saved when power is removed in EEPROM memory. With the EEPROM API functions, data (in byte) can conveniently be read and written to EEPROM.



Note

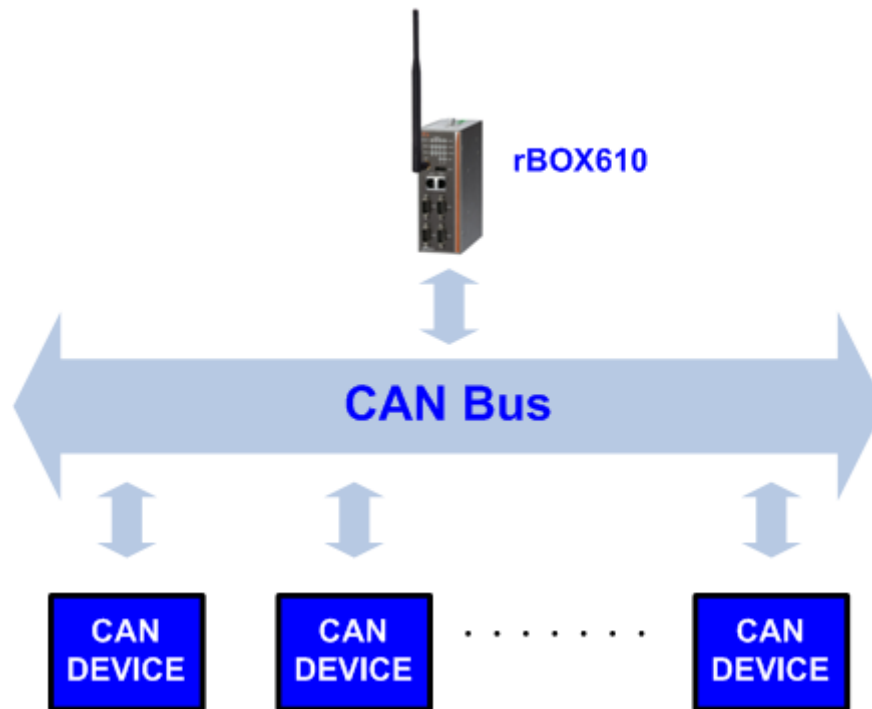
If you are going to use the rBOXAP, you are recommended not to change any data on EEPROM.

```
char str[] = "Hello world!!!";
printf("write \"Hello world!!!\" to EEPROM...\n");
for(i=0; i<sizeof(str); i++)
Write_EEPROM_Byte(i, str[i]);

printf("Show EEPROM ...\n");
for(i=0; i<=0xff; i++) {
    printf("%02x ", Read_EEPROM_Byte(i));
    if((i+1)%16 == 0)
        printf("\n");
}
```

4.3 CAN Bus

The Controller Area Network (CAN) bus is a serial bus protocol that usually used in connecting intelligent industrial device networks and building smart automatic control systems. Use the SocketCAN API to read and write to CAN bus on rBOX610. An example program showing how it works is provided below.



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <net/if.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <linux/can.h>
#include <linux/can/raw.h>
```

```
int main(void)
{
    int s,s1;
    int nbytes;
    struct sockaddr_can addr,addr1;
    struct can_frame frame,frame1;
    struct ifreq ifr,ifr1;
    int xBitRate=500000;

    char *ifname = "can0";
    char *ifname1 = "can1";
```



```

if((s = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
    perror("Error while opening socket");
    return -1;
}
if((s1 = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
    perror("Error while opening socket");
    return -1;
}

strcpy(ifr.ifr_name, ifname);
strcpy(ifr1.ifr_name, ifname1);
ioctl(s, SIOCGIFINDEX, &ifr);
ioctl(s1, SIOCGIFINDEX, &ifr1);

addr.can_family = AF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;
ifr.ifr_ifru.ifru_ival = xBitRate;
ioctl(s, SIOCSCANBAUDRATE, &ifr);

addr1.can_family = AF_CAN;
addr1.can_ifindex = ifr1.ifr_ifindex;
ifr1.ifr_ifru.ifru_ival = xBitRate;
ioctl(s1, SIOCSCANBAUDRATE, &ifr1);

printf("%s at index %d\n", ifname, ifr.ifr_ifindex);
printf("%s at index %d\n", ifname1, ifr1.ifr_ifindex);

if(bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
    perror("Error in socket bind");
    return -2;
}
if(bind(s1, (struct sockaddr *)&addr1, sizeof(addr1)) < 0) {
    perror("Error in socket bind");
    return -2;
}

frame.can_id = 0x123;
frame.can_dlc = 2;
frame.data[0] = 0x11;
frame.data[1] = 0x22;

write(s, &frame, sizeof(struct can_frame));
printf("Wrote data[0]:%2x,data[1]:%2x\n", frame.data[0], frame.data[1]);
read(s1, &frame1, sizeof(struct can_frame));
printf("Read data[0]:%2x,data[1]:%2x\n", frame1.data[0], frame1.data[1]);

return 0;
}

```

4.4 Compile Demo Program

4.4.1 Install LTIB Toolchain

Before you develop and compile sample program, you should install Linux toolchain into development PC. To do so, refer to Chapter 5 Board Support Package.

To compile and build demo program for rBOX610, please do:

Change to *ltib* directory.

```
$ cd ~/Project/Q7M100/ltib
```

Enter ltib shell mode (this is a developer function that provides an environment for compiling and building package).

```
$ ./ltib -m shell
```

Extract driver source to *ltib/rpm/BUILD* directory.

```
LTIB> tar jxf rBOX610-rb_lib-1.1.1.tar.bz2 -C rpm/BUILD/
```

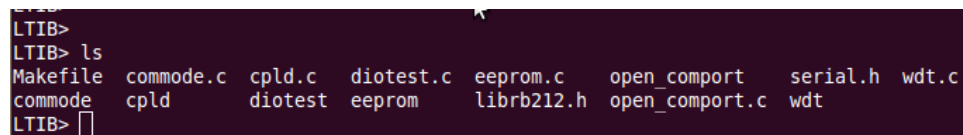
Change to *rb_lib/demo* directory.

```
LTIB> cd rpm/BUILD/rb_lib/demo
```

Build the demo program.

```
LTIB> make
```

Then you should have example programs such as *open_comport*, *wdt*, *cpld*, *diotest*, *eeeprom* and *commode*.



```
LTIB>
LTIB> ls
Makefile  commode.c  cpld.c    diotest.c  eeeprom.c  open_comport  serial.h  wdt.c
commode   cpld       diotest   eeeprom    librb212.h  open_comport.c  wdt
LTIB> 
```

4.4.2 Run demo program

Refer to section 2.3 for detailed information.

Chapter 5

Board Support Package (BSP)

5.1 Host Development System Installation

5.1.1 Install Host System

1. Download Ubuntu 10.04 LTS iso image.
2. Install Ubuntu 10.04.
3. Sudoers:
To edit the sudoer's file, please run 'sudo visudo'. At the end of the sudoers file, add the following line; which is needed for using LTIB. This assumes that all your developers have administrator privileges on this host. If that is not the case, add a similar line for each user.

```

jrtiger@jrtiger-laptop: ~/Projects/i.MX28/ltib
File Edit View Terminal Help
GNU nano 2.2.2 File: /etc/sudoers.tmp

# (Note that later entries override this, so you might need to move
# it further down)
%sudo ALL=(ALL) ALL
#
#includedir /etc/sudoers.d

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
jrtiger ALL = NOPASSWD: ALL
%admin ALL = NOPASSWD: /usr/bin/rpm, /opt/freescale/ltib/usr/bin/rpm

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

4. Install host packages needed by LTIB as follows:

```

$ sudo aptitude -y install gettext libgtk2.0-dev rpm bison m4 libfreetype6-dev
$ sudo aptitude -y install libdbus-glib-1-dev liborbit2-dev intltool
$ sudo aptitude -y install ccache ncurses-dev zlib1g zlib1g-dev gcc g++ libtool
$ sudo aptitude -y install uuid-dev liblz2-dev
$ sudo aptitude -y install tcl

```

5. Install and configure TFTP server:
After tftpd is installed, configure it by editing `/etc/xinetd.d/tftp`. Change the default export path (it is either `/usr/var/tftpboot` or `/var/lib/tftpboot`) to `/`. Or change the default export path to whatever directory you want to download from. Then reboot the hardware.

```
$ sudo aptitude -y install tftp tftpd xinetd
```

```
$ sudo vi /etc/xinetd.d/tftp
```

```

jrtiger@jrtiger-laptop: ~/script
File Edit View Terminal Help
Service tftp
{
    socket_type      = dgram
    protocol         = udp
    wait             = yes
    user             = root
    disable          = no
    server           = /usr/sbin/in.tftpd
    server_args      = -s /tftpboot
}

"/etc/xinetd.d/tftp" [readonly] 10L, 253C 1,1 All

```

Then restart the TFTP server.

```
$ sudo /etc/init.d/xinetd restart
```

6. Install and configure NFS server:
\$ sudo aptitude -y install nfs-common nfs-kernel-server portmap
To configure nfs server, add lines to `/etc/exports` as follows:
`/tools/rootfs *(rw, sync, no_root_squash)`
\$ sudo vi /etc/exports
Create a symbolic link to root filesystem which your ltib build.
\$ sudo mkdir /tools
\$ sudo ln -s ~/Project/Q7M100/ltib/rootfs /tools/rootfs

Then restart the NFS server.

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

5.1.2 Install LTIB

1. Build your own project folder.
`$ mkdir -p ~/Project/Q7M100`
2. Extract the source tar ball with the following command:
`$ tar xzf L2.6.35_10.12.01_SDK_source.tar.gz`

Note that you can get this file from Axiomtek official website:

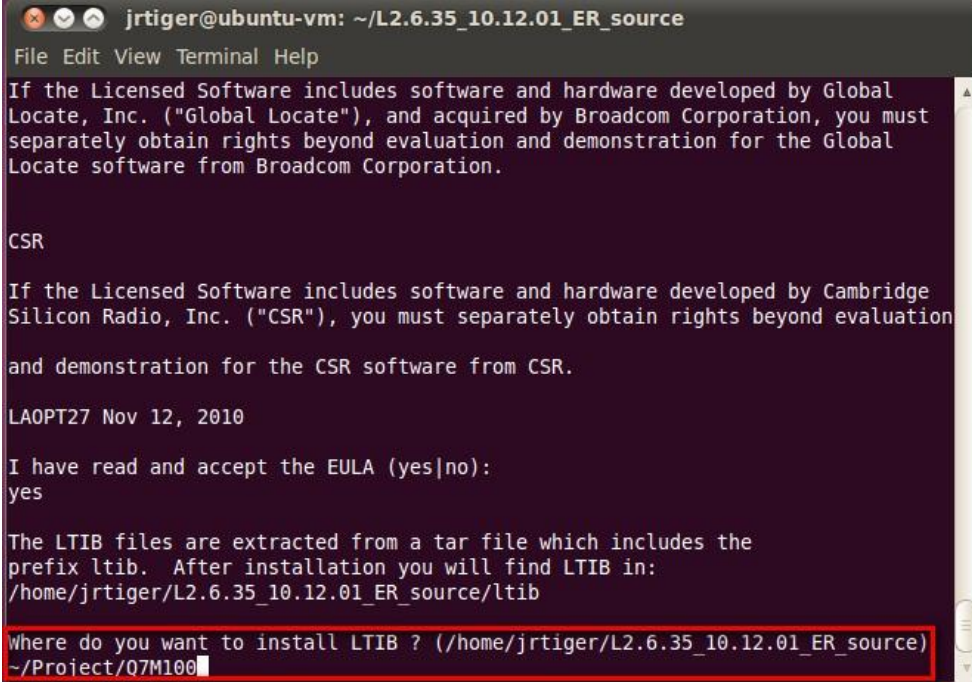
<http://www.axiomtek.com/products/ViewDownload.asp?View=PID-rBOX610>

Download LTIB_IMX28.zip

Unzip it to get 2.6.35_10.12.01_SDK_docs.tar.gz,

L2.6.35_10.12.01_SDK_source.tar.gz and IMX_MMCODECS_10.12.tar.gz .

3. Now change directory to the extracted folder and execute the install script. Then you can install it into your own project folder.
`$ cd L2.6.35_10.12.01_ER_source`
`$./install`



```
jrtiger@ubuntu-vm: ~/L2.6.35_10.12.01_ER_source
File Edit View Terminal Help
If the Licensed Software includes software and hardware developed by Global
Locate, Inc. ("Global Locate"), and acquired by Broadcom Corporation, you must
separately obtain rights beyond evaluation and demonstration for the Global
Locate software from Broadcom Corporation.

CSR

If the Licensed Software includes software and hardware developed by Cambridge
Silicon Radio, Inc. ("CSR"), you must separately obtain rights beyond evaluation
and demonstration for the CSR software from CSR.

LAOPT27 Nov 12, 2010

I have read and accept the EULA (yes|no):
yes

The LTIB files are extracted from a tar file which includes the
prefix ltib. After installation you will find LTIB in:
/home/jrtiger/L2.6.35_10.12.01_ER_source/ltib

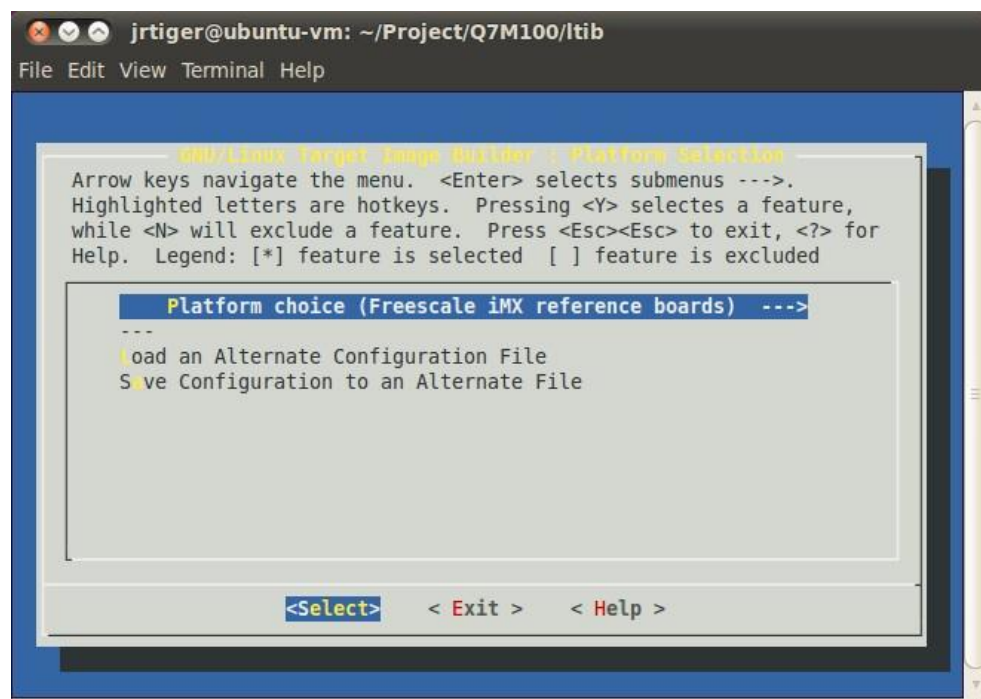
Where do you want to install LTIB ? (/home/jrtiger/L2.6.35_10.12.01_ER_source)
~/.Project/Q7M100
```

4. Configure and build.
This command invokes LTIB with its default behavior of performing a build. Since the installation has not yet been configured, LTIB will present the configuration screen before building. Then, when you exit the configuration screen, LTIB will build the target image. Also, at the very first time you run LTIB on a system, it will build and install a number of host tools that it will place under `/opt/freescale`.

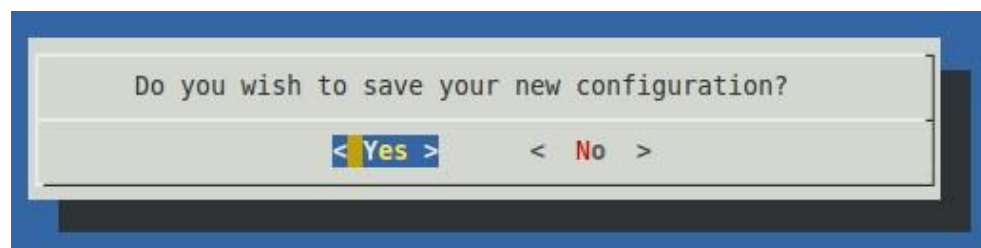
To ask LTIB to show the configuration screen without building afterwards, run this command:

```
$ ./ltib -m config
```

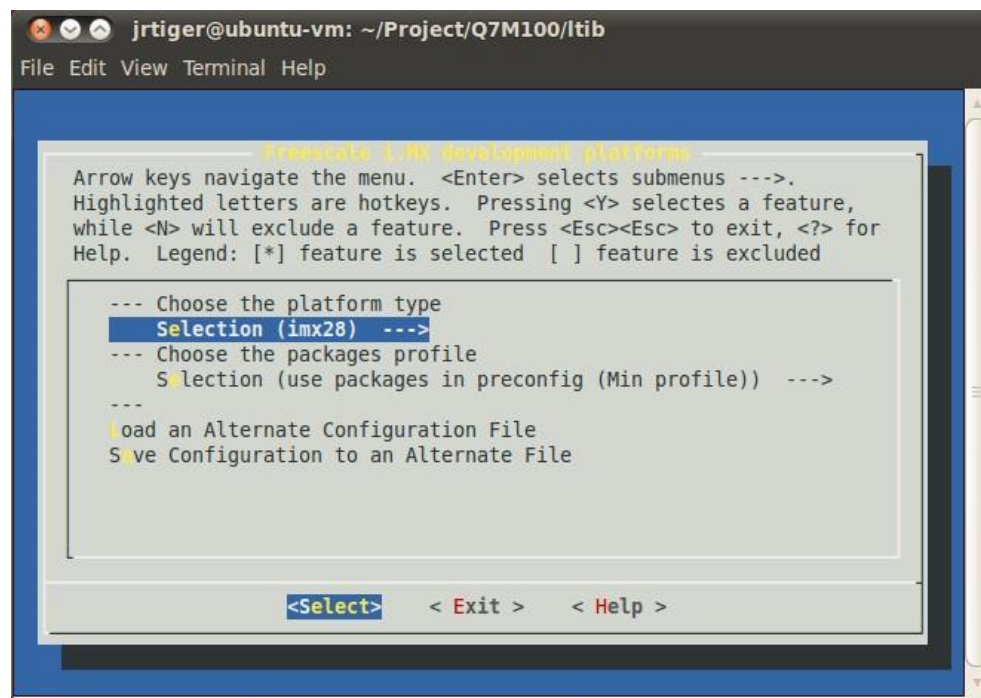
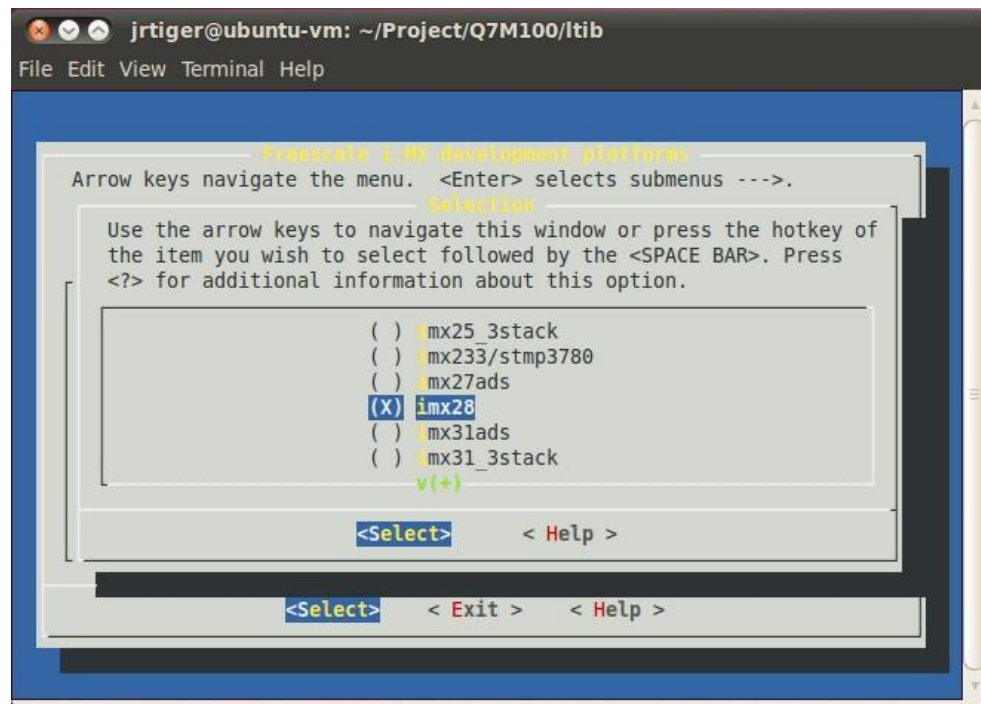
5. First time configuration; please select the iMX platform.



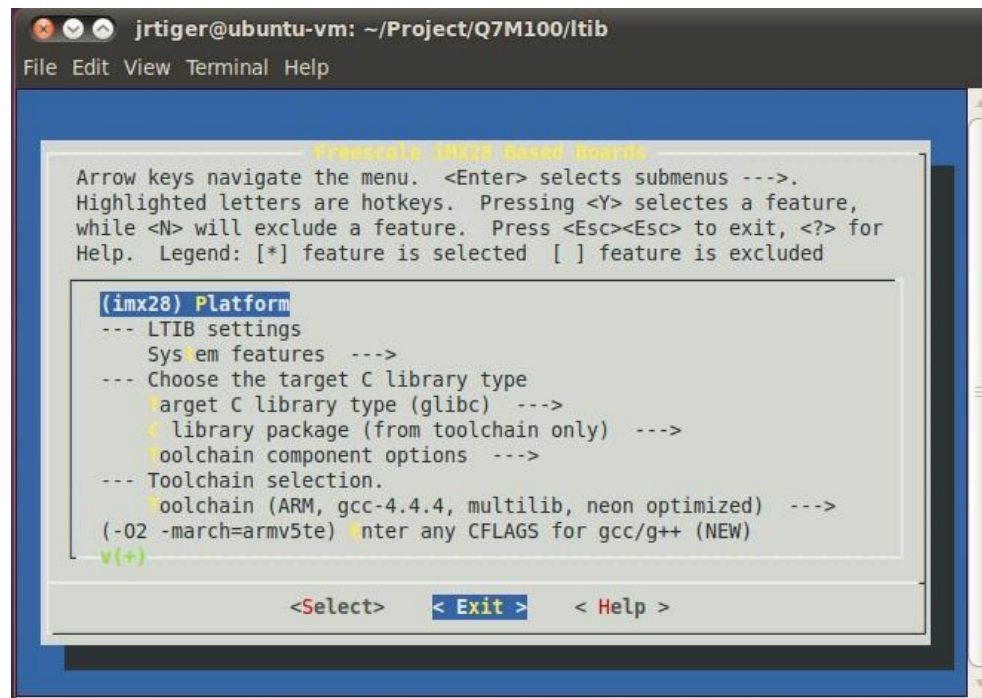
Then save the configuration.



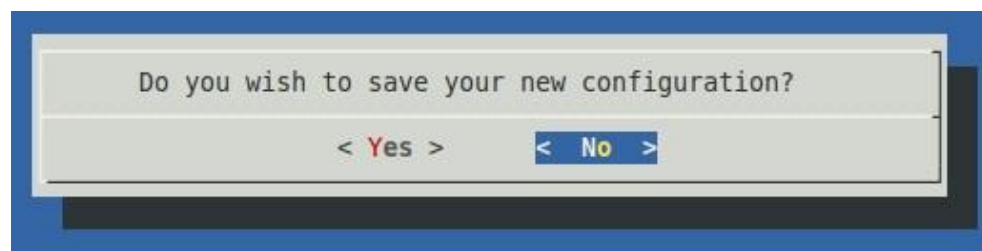
Select the imx28 sub-platform.



Exit the sub-platform selection screen and save the configuration. This will bring you to the main LTIB configuration screen.



Just exit without saving main LTIB configuration.



6. Create and patch kernel for rBOX610.
Create the local kernel folder.
\$ mkdir -p ~/Project/Q7M100/kernel

Extract kernel source to this folder.

```
$ tar jxf /opt/freescale/pkgs/linux-2.6.35.3.tar.bz2 -C ~/Project/Q7M100/kernel
```

Extract i.MX platform patches to kernel source folder.

```
$ tar jxf /opt/freescale/pkgs/linux-2.6.35.3-imx_10.12.01.bz2 -C
~/Project/Q7M100/kernel/linux-2.6.35.3
```

Enter the kernel source folder and patch the patches.

```
$ cd ~/Project/Q7M100/kernel/linux-2.6.35.3/
$ ./patches/patch-kernel.sh
```

Patch the rBOX610 patches which you can copy from our BSP package.

```
$ patch -p1 < ~/patch-2.6.35.3-Q7M100-016.patch
```

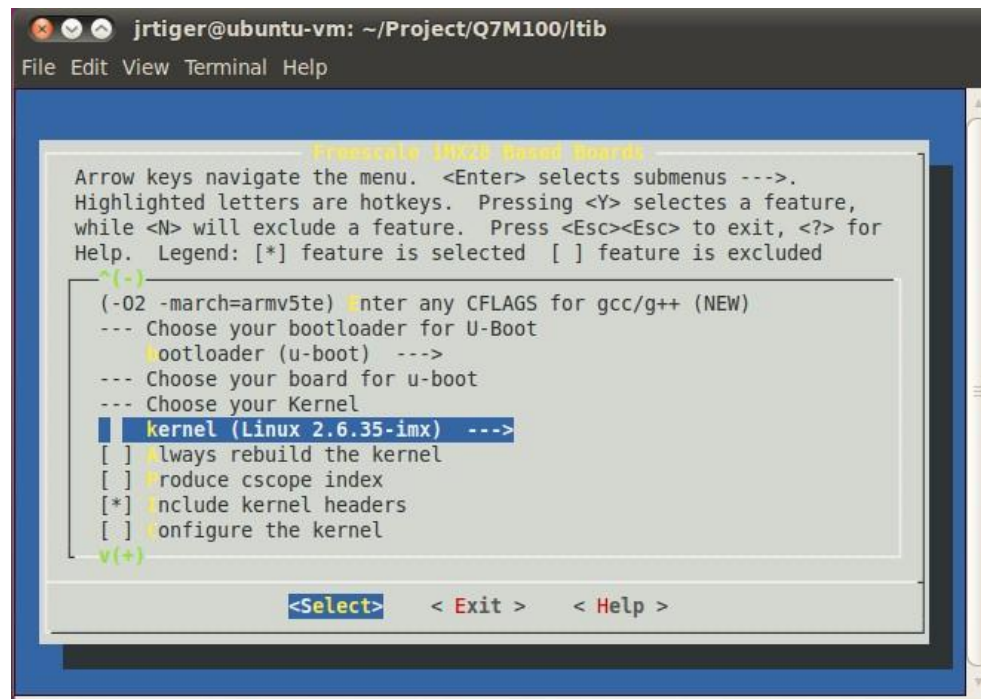

Return to `~/Project/Q7M100/kernel` folder and make a symbolic link to kernel source folder.

```
$ cd ~/Project/Q7M100/kernel
$ ln -s linux-2.6.35.3 linux
```

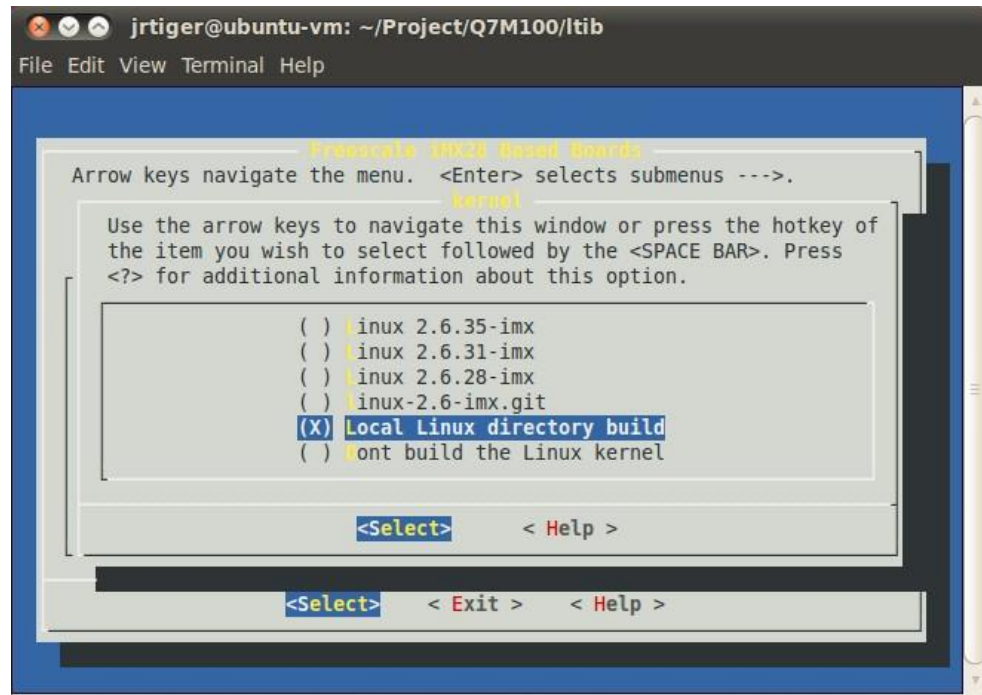
7. Reconfigure LTIB main setting to build kernel to local Linux directory.

```
$ cd ~/Project/Q7M100/ltib
$ ./ltib -m config
```

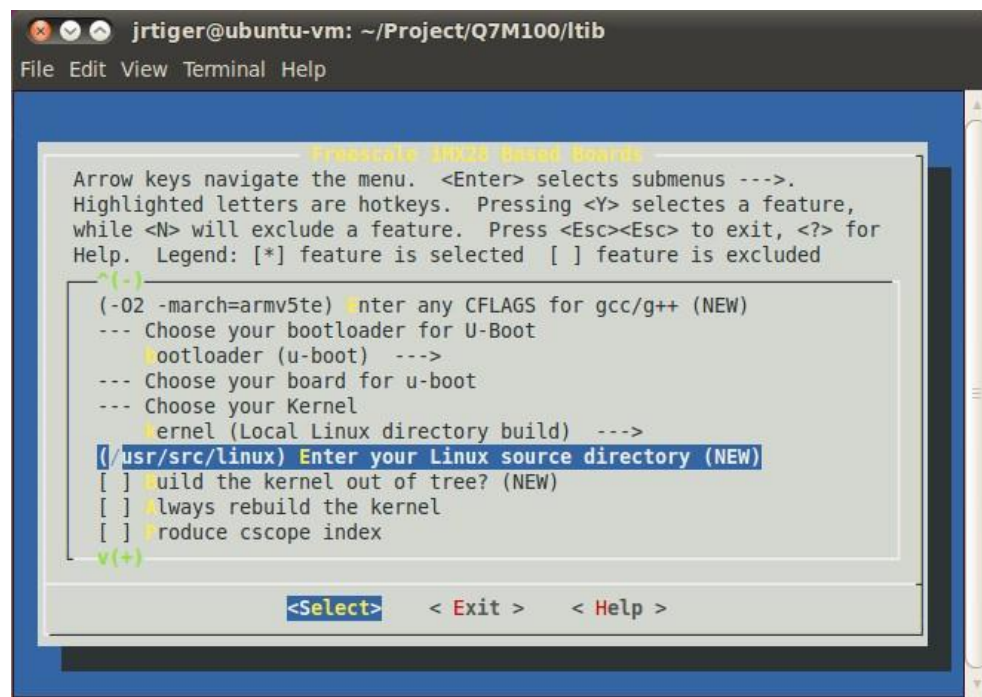
Select kernel setting.



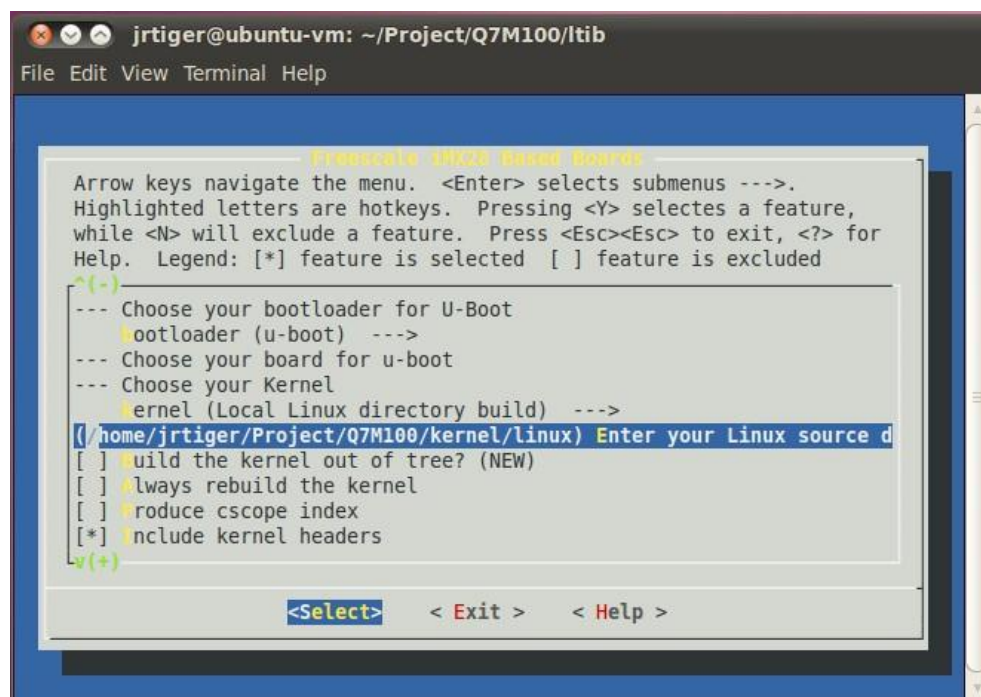
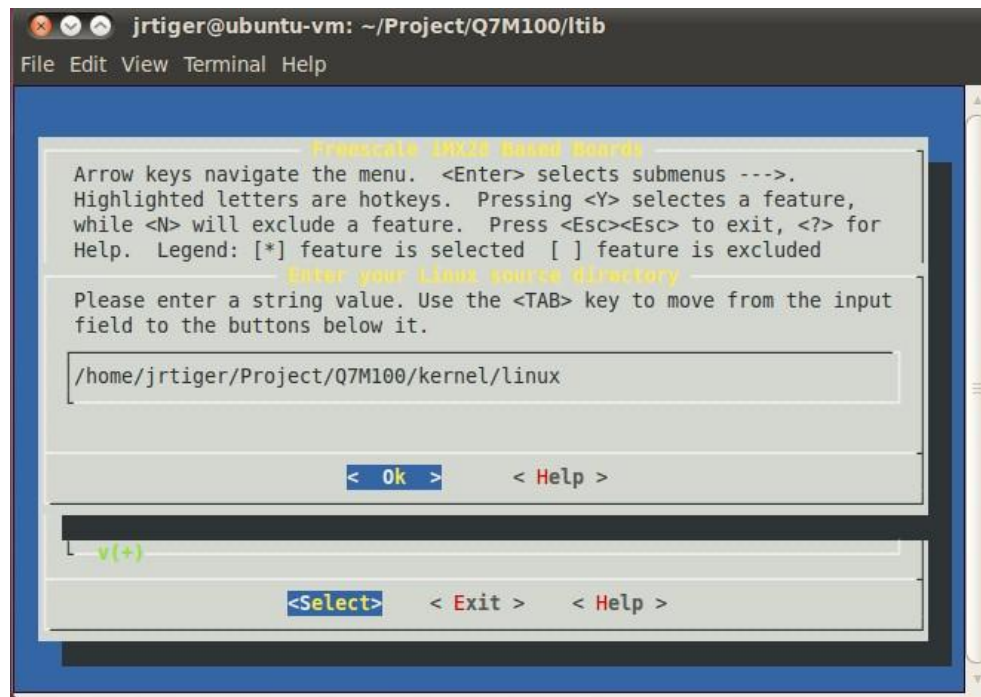
Select Local Linux directory build.



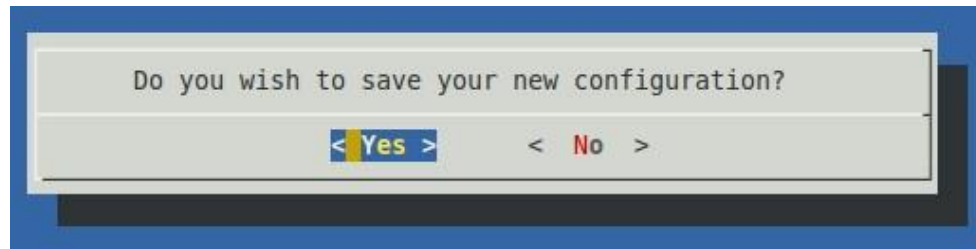
Then enter new linux source directory.



Enter your own linux kernel source folder.



Finally save the configuration.



8. Options:

Adding iMX28 Multimedia codecs support:

Note that you can get this file from Axiomtek official website:

<http://www.axiomtek.com/products/ViewDownload.asp?View=PID-rBOX610>

Download LTIB_IMX28.zip

Unzip it to get 2.6.35_10.12.01_SDK_docs.tar.gz,

L2.6.35_10.12.01_SDK_source.tar.gz and IMX_MMCODECS_10.12.tar.gz .

Extract codec file.

```
$ tar xzf IMX_MMCODECS_10.12.tar.gz
```

Copy all tar.gz file to /opt/freescale/pkgs folder.

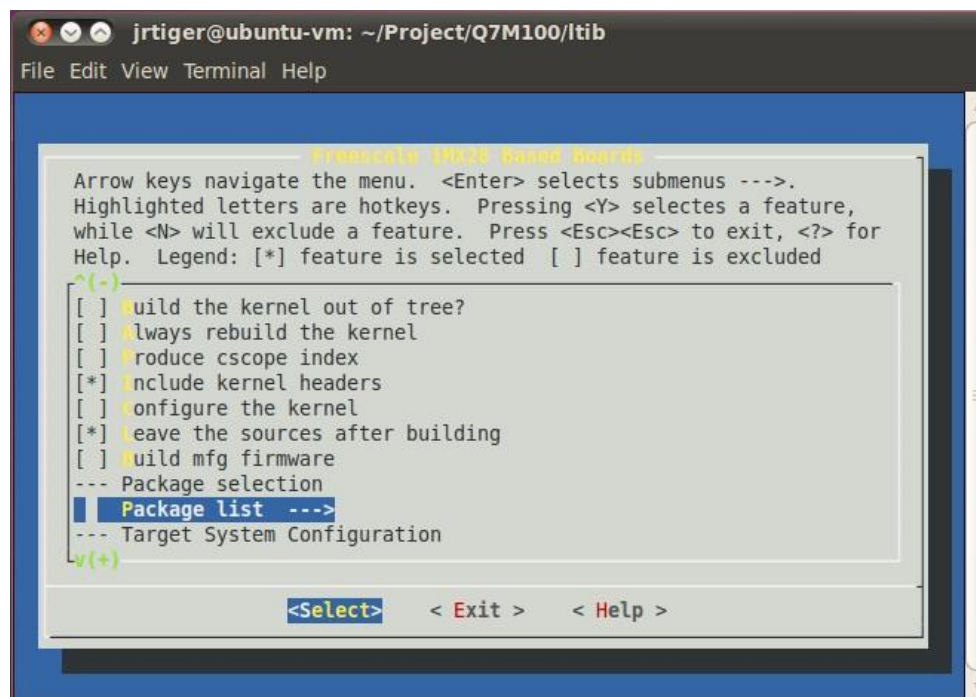
```
$ cp IMX_MMCODECS_10.12/*.tar.gz /opt/freescale/pkgs/
```

Then reconfigure LTIB main setting.

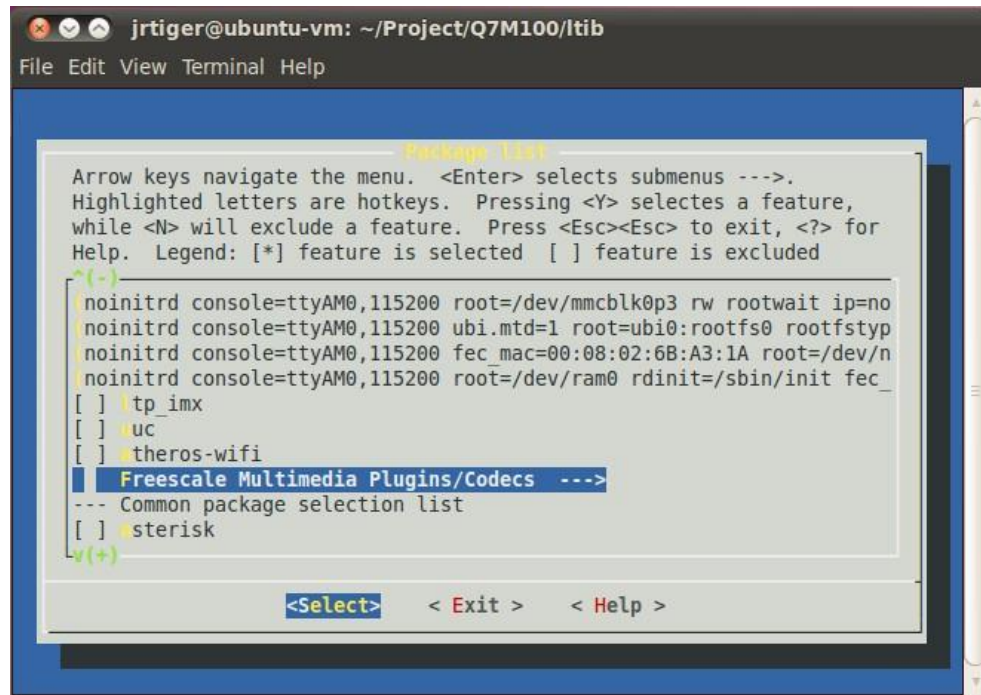
```
$ cd ~/Project/Q7M100/ltib
```

```
$ ./ltib -m config
```

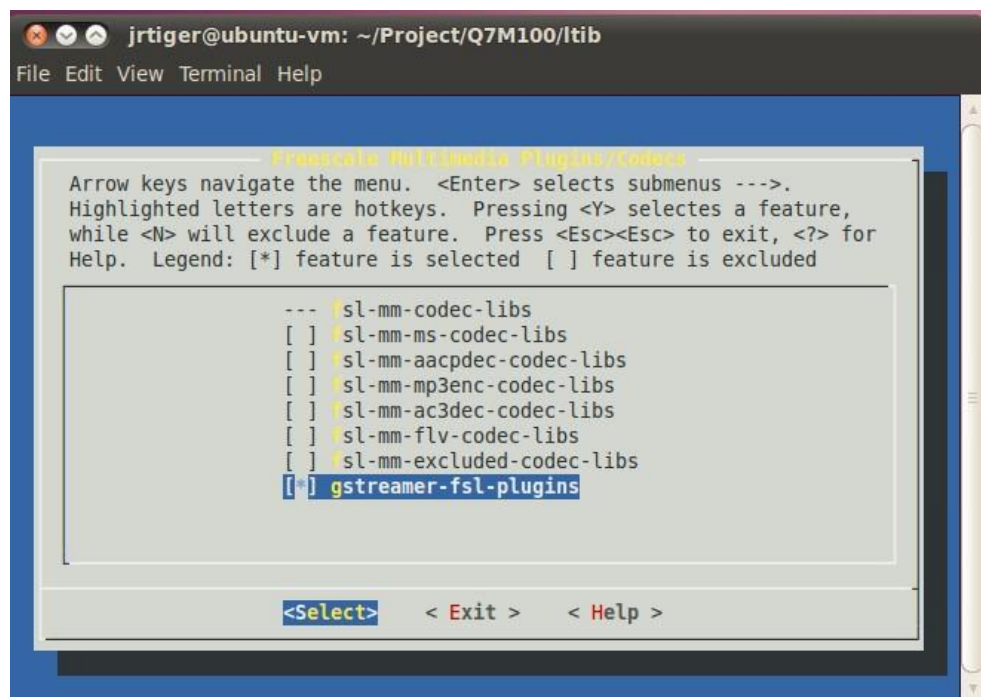
Select Package list.



Select Freescale Multimedia Plugins/Codecs.

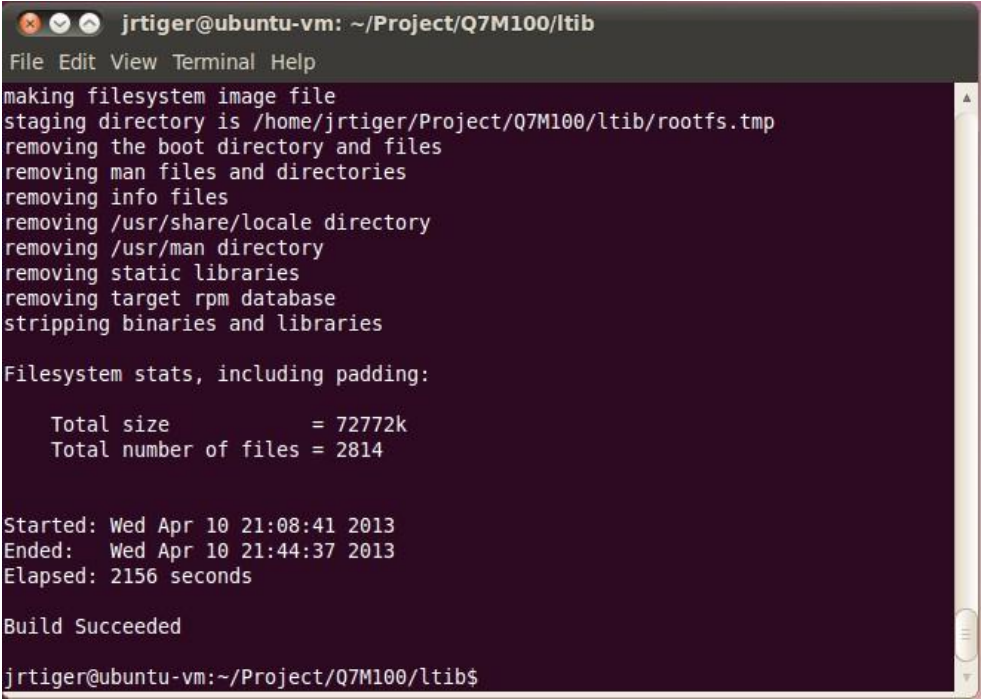


Select codecs you want to add.



9. Build the target kernel image and root filesystem.

```
$ cd ~/Project/Q7M100/ltib  
$ ./ltib
```



```
jrtiger@ubuntu-vm: ~/Project/Q7M100/ltib  
File Edit View Terminal Help  
making filesystem image file  
staging directory is /home/jrtiger/Project/Q7M100/ltib/rootfs.tmp  
removing the boot directory and files  
removing man files and directories  
removing info files  
removing /usr/share/locale directory  
removing /usr/man directory  
removing static libraries  
removing target rpm database  
stripping binaries and libraries  
  
Filesystem stats, including padding:  
  
    Total size          = 72772k  
    Total number of files = 2814  
  
Started: Wed Apr 10 21:08:41 2013  
Ended:   Wed Apr 10 21:44:37 2013  
Elapsed: 2156 seconds  
  
Build Succeeded  
  
jrtiger@ubuntu-vm:~/Project/Q7M100/ltib$
```

After you have completed a build using LTIB, you will have a target root filesystem in the *rootfs* directory inside the LTIB install directory.

Inside the *~/Project/Q7M100/ltib/rootfs/boot* directory, you can find kernel image *ulmage*.

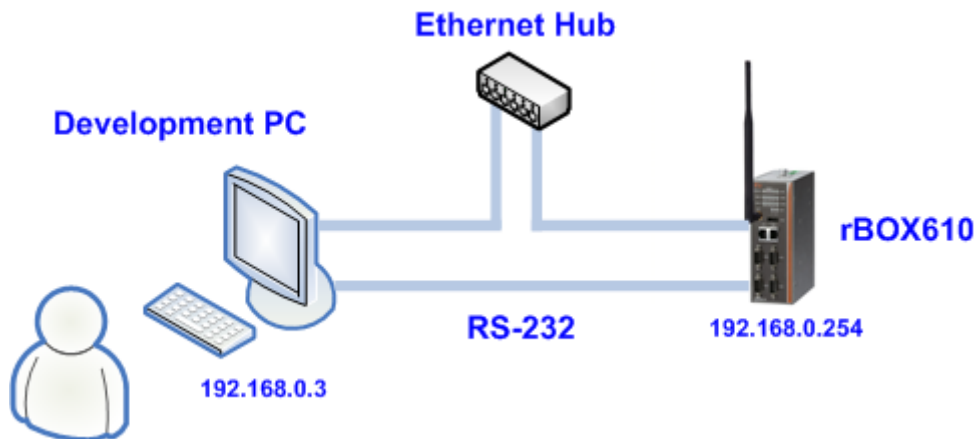
5.1.3 Compile Demo Program

1. Compile and build demo program for rBOX610 CPLD function.
Change to *ltib* directory.
`$ cd ~/Project/Q7M100/ltib`
Enter ltib shell mode (this is a developer function that provides an environment for compiling and building package).
`$./ltib -m shell`
Extract driver source to *ltib/rpm/BUILD* directory.
`LTIB> tar jxf rBOX610-rb_lib-1.1.1.tar.bz2 -C rpm/BUILD/`
Change to *rb_lib/demo* directory.
`LTIB> cd rpm/BUILD/rb_lib/demo`
Build the demo program.
`LTIB> make`
Then copy all binary file to target root filesystem you build. The root filesystem is a directory tree found under *rootfs*.
2. Install and run demo program into rBOX610.
Refer to section 2.3 for more detailed information.

5.2 U-Boot for Q7M100

5.2.1 Booting the System with an NFS Filesystem

By default, U-Boot is configured to boot from NFS. To boot from NFS, first you must set some configurations. Press any key to break from the boot progress and set configurations.



```
Setup TFTP server IP:  
MX28 U-Boot > setenv serverip 192.168.0.3  
Setup board IP address:  
MX28 U-Boot > setenv ipaddr 192.168.0.254  
Setup rootfs path:  
MX28 U-Boot > setenv nfsroot /tools/rootfs  
Setup boot delay:  
MX28 U-Boot > setenv bootdelay 2  
Save configurations:  
MX28 U-Boot > saveenv
```

Reset the board and kernel will be launched.

5.2.2 Booting the System from eMMC

First you need to download kernel root filesystem through TFTP server, and install it into eMMC.

```
Setup TFTP server IP:  
MX28 U-Boot > setenv serverip 192.168.0.3  
Setup board IP address:  
MX28 U-Boot > setenv ipaddr 192.168.0.254  
Download kernel image through TFTP server  
MX28 U-Boot > tftpboot ulmage  
Using FEC0 device  
TFTP from server 192.168.0.3; our IP address is 192.168.0.254  
Filename 'ulmage'.  
Load address: 0x42000000  
Loading: FEC: Link is down 7809
```



```
#####
done
Bytes transferred = 2454628 (257464 hex)
Install kernel image into eMMC:
MX28 U-Boot > mxs_mmc install 0 0x42000000 0x257464 ulmage
Done: 4795 (12bb hex) sectors written at 256 (100 hex)
```

Download root filesystem through TFTP server (the file size limitation of this method is 96MB):

```
MX28 U-Boot > tftpboot rootfs.ext2
Using FEC0 device
TFTP from server 192.168.0.3; our IP address is 192.168.0.254
Filename 'rootfs.ext2'.
Load address: 0x42000000
Loading: FEC: Link is down 7809
```

```
#####
done
Bytes transferred = 74526720 (4713000 hex)
Install root filesystem into eMMC:
MX28 U-Boot > mxs_mmc install 0 0x42000000 0x4713000 rootfs
Done: 145560 (23898 hex) sectors written at 32768 (8000 hex)
Run system from eMMC:
MX28 U-Boot > run bootcmd_mmc
```



Note

We configure 1GB EXT partition for root filesystem. If you want to use full size to deploy your system, try booting from NFS filesystem and then format EXT partition. Extract filesystem and install it to EXT partition as follows:

Booting with NFS filesystem:
MX28 U-Boot > run bootcmd_net
Format EXT partition as EXT3:
\$ mkfs.ext3 /dev/mmcblk0p3
Mount this partition:
\$ mount /dev/mmcblk0p3 /mnt/src
Extract filesystem and install it to EXT partition:
\$ tar jxf rootfs.tar.bz2 -C /mnt/src
Then reboot and boot it from eMMC.

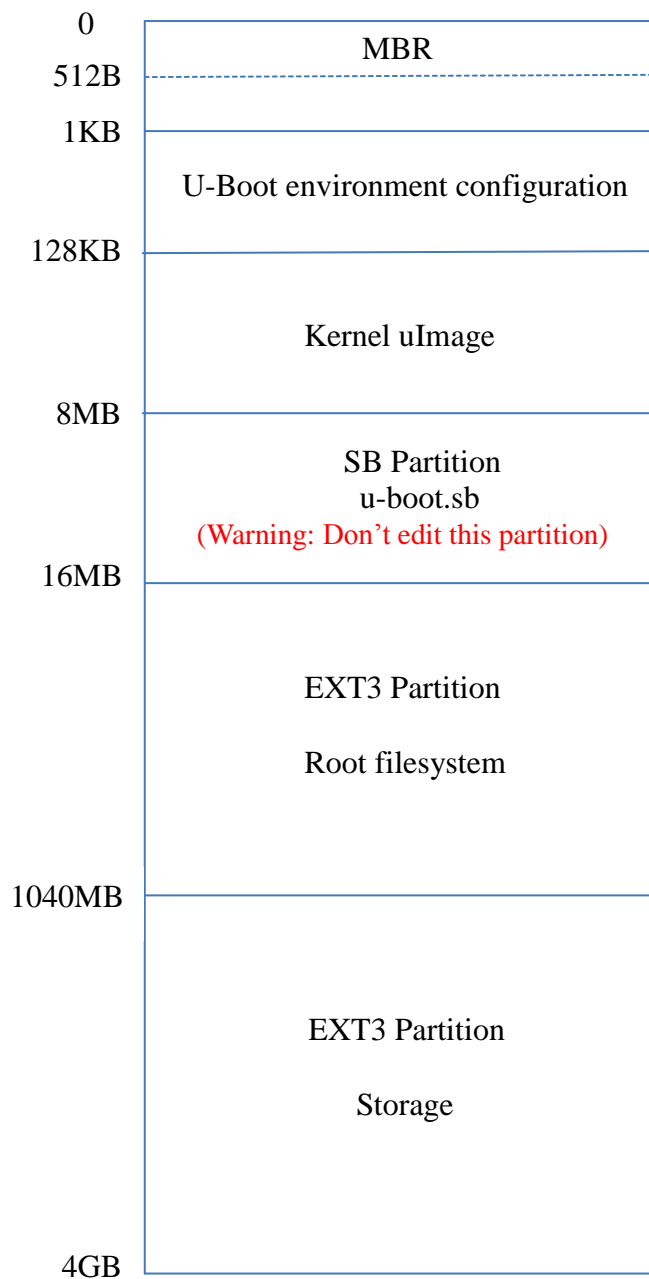
Command syntax of installing image to eMMC:
mxs_mmc install <dev num> addr size [ulmage | rootfs]

5.2.3 Reference Document

- uboot_mx28.pdf
- ltib_build_host_setup.pdf

5.3 Additional Information

5.3.1 4GB eMMC Partition Layout



5.3.2 Compile and Build Your Program

1. Compile and build your program for rBOX610.
Change to *ltib* directory.
`$ cd ~/Project/Q7M100/ltib`
Enter ltib shell mode (this is a developer function that provides an environment for compiling and building package).
`$./ltib -m shell`
And compile your program source code.
`LTIB> make`

Then copy all binary file to target root filesystem you build. The root filesystem is a directory tree found under *rootfs*.

2. Install your program into rBOX610.
Through Ethernet:
You can use 'tftp' or 'ftpget' command to download it to EXT3 partition such as */root*, */mnt/storage* or other folders.
Through USB storage:
Mount your USB device and copy it to the folder.

Refer to section 2.3 for more detailed information.

This page is intentionally left blank.

Chapter 6

Application Software

The rBOX610 comes with an application software package containing various components such as Serial Server, Modbus Gateway, 3G Configuration, GPS Configuration, DIO Configuration, SNMP, Alarm, and etc. Users also can download this package from website given below and install it on rBOX610.

<http://www.axiomtek.com/Download/Download/rBOX610/rBOX610AP-2.2.2.tar.gz>

6.1 Install the Application

Follow steps below to install the application package on rBOX610.

1. Download rBOX610AP-2.2.2.tar.gz.
2. To extract the compressed file.
For Windows® users, you may use the WinRAR compression software utility.
For Linux users, extract the source tar ball with the following command:
`$ tar xzf rBOX610AP-2.2.2.tar.gz`
3. Now you can see the following 3 files:
 - rBOX610AP: Application package installation program.
 - uninstall.sh: A script file for uninstalling application package.
 - readme: A readme file.
4. Copy rBOX610AP and uninstall.sh to USB flash drive.
5. Boot up the rBOX610 and mount USB flash drive.
`$ mkdir -p /mnt/usb`
`$ mount /dev/sda1 /mnt/usb`
`$ cp /mnt/usb/rBOX610AP /root`
`$ cp /mnt/usb/uninstall.sh /root`

Install the application.

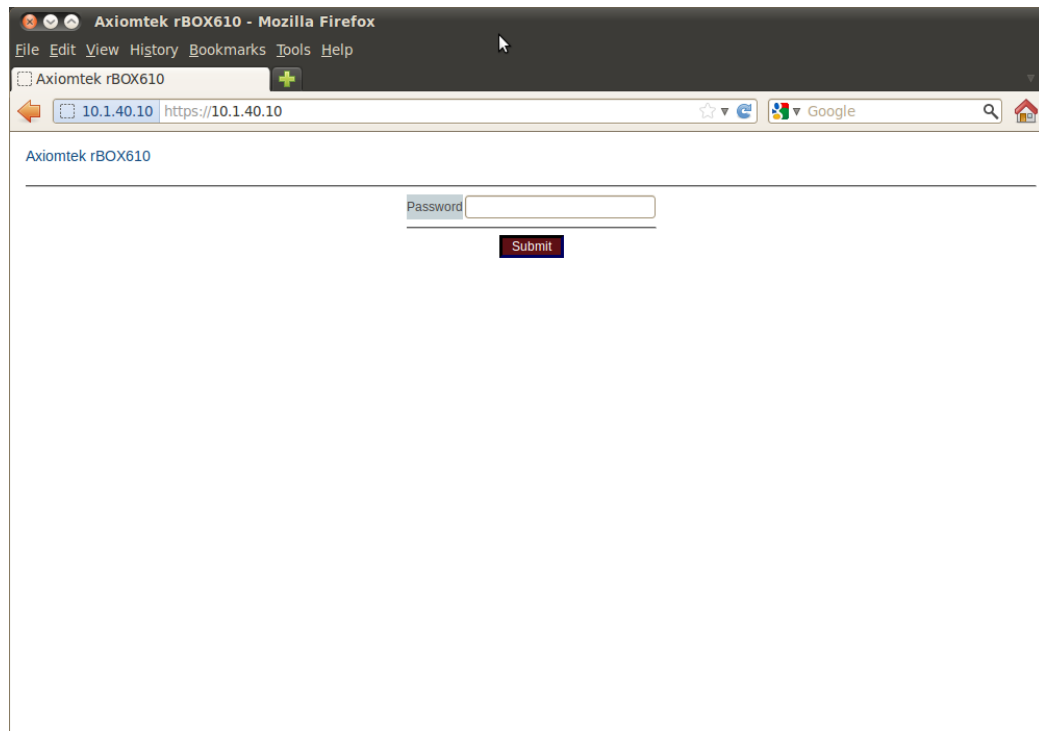
```
$ cd /root
```

```
$ ./rBOX610AP
```

If you see the message 'Install success', then please reboot rBOX610.

```
$ reboot
```

6. After successful installation, the App LAN1 default IP address is 192.168.0.254 and LAN2 default IP address is 192.168.10.1. Open a web browser and key in the IP address into web browser's address bar. Then you can see the following web.config page displayed.



Note

For more detailed information about the Web and App configuration, please refer to the rBOX610-FL Web and App User's Manual.

6.2 Uninstall the Application

Follow steps below to uninstall the application package from rBOX610.

1. Execute `uninstall.sh` to uninstall the application.

```
$ cd /root  
$ sh uninstall.sh
```
2. Reboot rBOX610.

```
$ reboot
```