DEVELOPMENT GUIDE

VAB-820-X

Linux BSP v1.0

1.02-01142015-161100

Copyright

Copyright © 2015 VIA Technologies Incorporated. All rights reserved.

No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise without the prior written permission of VIA Technologies, Incorporated.

Trademarks

All brands, product names, company names, trademarks and service marks are the property of their respective holders.

Disclaimer

VIA Technologies makes no warranties, implied or otherwise, in regard to this document and to the products described in this document. The information provided in this document is believed to be accurate and reliable as of the publication date of this document. However, VIA Technologies assumes no responsibility for the use or misuse of the information in this document and for any patent infringements that may arise from the use of this document. The information and product specifications within this document are subject to change at any time, without notice and without obligation to notify any person of such change.

VIA Technologies, Inc. reserves the right the make changes to the products described in this manual at any time without prior notice.

Revision History

Version	Date	Remarks
1.0	9/4/2014	Initial release
1.1	11/27/2014	Modified Section 3.2.2 Modified Appendix A Added Appendix C Added Appendix D
1.2	1/12/2015	Modified Appendix A Added Appendix E Added Appendix F

Table of Contents

Lists of Figures

Lists of Tables

1. Introduction

The purpose of this document is to provide a practical introduction on developing software for the VAB-820-X (SoC core: Freescale i.MX6 DualLite) on a Linux development host only.

1.1. Overview

The VIA VAB-820-X platforms are embedded systems powered by ARM processor with Linux kernel 3.0.35 operating system by default. Major functions of the Linux include all system-requirement shell commands and drivers ready for VAB-820-X platform. The Solution package VAB-820-X does not offer a development environment. Users can develop it under an Ubuntu environment.

There are three major boot components for Linux, the **"u-boot.bin", "uImage"** and **"Root File System"**. The **"u-boot.bin"** is for initial peripheral hardware parameter. The **"uImage"** is the Linux kernel image, and the **"Root File System"** is for Linux O.S. The system will not boot successfully into a Linux environment if one of these files does not exist in the boot media (SPI ROM, SD storage card or onboard eMMC).

This development guide will use VAB-820 as an example instead of VAB-820-X to describe relational building procedure.

1.2. Package Content

There are three folders in VAB-820-X Linux BSP.

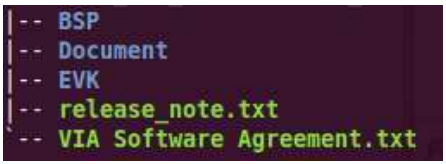


Figure 1. VAB-820-X Linux BSP content

1.2.1. BSP Folder Contents

- **LTIB (Linux Target Image Builder):** A tool that can be used to develop and deploy BSPs (Board Support Packages) for a number of embedded target platforms including PowerPC, ARM.
- **PatchFiles:** This folder provides u-boot/kernel patch files for VAB-820.

1.2.2. EVK Folder Contents

- **vab-820_demo_image.tar.bz2:** Configure files when user would like to evaluate VAB-820 with Ubuntu root file system.

Note:
If a user needs the supporting files for all software mentioned in VAB-820-X Linux BSP document, please contact our regional sales representative for assistance.

2. Setup Building Environment

This chapter will guide you through setting up your developing environment. All instructions in this guide are for Ubuntu 10.04 (32Bit). Please install the Ubuntu to your PC/NB in advance.

2.1. Configure Ubuntu

2.1.1. Change Default Editor (optional)

The default editor is NANO. To set vi as the default editor:

```
user@ubuntu:~$ sudo update-alternatives --config editor
There are 3 choices for the alternative editor (providing /usr/bin/editor).

  Selection    Path               Priority   Status
------------------------------------------------------------
* 0            /bin/nano           40        auto mode
  1            /bin/ed            -100       manual mode
  2            /bin/nano           40        manual mode
  3            /usr/bin/vim.tiny   10        manual mode

Press enter to keep the current choice[*], or type selection number: 3
update-alternatives: using /usr/bin/vim.tiny to provide /usr/bin/editor
(editor) in manual mode.
```

2.1.2. Sudoers

The sudoer's file must be updated to allow the login account to run the rpm commands as root. The login for this example is **"user"**. Edit the sudoer's file using the **$sudo visudo** and add the line after the comment **"#User alias specification"** as shown below. The first word **"%user"** will be the login name you are using. Save the changes when it is done.

```
user@ubuntu:~$ sudo visudo
...
...
# User alias specification
%user ALL = NOPASSWD: /usr/bin/rpm, /opt/freescale/ltib/usr/bin/rpm
```

2.1.3. Install the Host Packages

The following packages are installed to support a LTIB development environment and presented in a bash script that can be cut and pasted into your environment and execute:

```
#/bin/bash

# Install LTIB dependant packages
sudo apt-get install gettext libgtk2.0-dev rpm bison m4 libfreetype6-dev
sudo apt-get install libdbus-glib-1-dev liborbit2-dev intltool
sudo apt-get install ccache ncurses-dev zlib1g zlib1g-dev gcc g++ libtool
sudo apt-get install uuid-dev liblzo2-dev
sudo apt-get install tcl dpkg

sudo apt-get install texinfo texlive

# The following recommended for Linux development.
# They are not required by LTIB.
sudo apt-get install gparted openssh-server
sudo apt-get install nfs-common nfs-kernel-server lintian
sudo apt-get install git-core git-doc git-email git-gui gitk
sudo apt-get install diffstat indent tofrodos fakeroot doxygen uboot-
mkimage
sudo apt-get install sendmail mailutils meld atftpd sharutils
sudo apt-get install manpages-dev manpages-posix manpages-posix-dev linux-
doc
sudo apt-get install vnc4server xvnc4viewer
```

2.1.4. Change the Default Shell

The Ubuntu default shell is dash. To change the default shell to bash, select **<No>** and exit. This will remove the dash and use bash.

```
user@ubuntu: $ sudo dpkg-reconfigure dash
```

Below is a screenshot to show the status before and after the reconfiguration.

```
user@ubuntu:~$ ls -l /bin/sh
lrwxrwxrwx 1 root root 4 2013-05-16 10:32 /bin/sh -> dash
user@ubuntu:~$ sudo dpkg-reconfigure dash
[sudo] password for user:
Removing 'diversion of /bin/sh to /bin/sh.distrib by dash'
Adding 'diversion of /bin/sh to /bin/sh.distrib by bash'
Removing 'diversion of /usr/share/man/man1/sh.1.gz to
/usr/share/man/man1/sh.distrib.1.gz by dash'
Adding 'diversion of /usr/share/man/man1/sh.1.gz to
/usr/share/man/man1/sh.distrib.1.gz by bash'
user@ubuntu:~$ ls -l /bin/sh
lrwxrwxrwx 1 root root 4 2013-05-16 10:32 /bin/sh -> bash
```

2.1.5. Configure Ccache (optional)

LTIB uses ccache to speed up the compilation. The cache that LTIB uses exists as a .ccache directory in each user's home directory. The path for this example is "/home/user/.ccache". This directory can grow to be quite large if no upper limit is set.

The followings are the ccache commands to limit the size to 50 MB. The size may change as you see fit, clean the **ccache** and show the settings.

```
[1] user@ubuntu:~$ ccache -M 50M
Set cache size limit to 50.0 Mbytes
[2] user@ubuntu:~$ ccache -c
Cleaned cache
[3] user@ubuntu:~$ ccache -s
cache directory                     /home/user/.ccache
cache hit (direct)                     0
cache hit (preprocessed)               0
cache miss                             0
files in cache                         0
cache size                             0 Kbytes
max cache size                      50.0 Mbytes
```

2.1.6. Change Permissions on /opt

The LTIB installation process creates the directory **"/opt/freescale"**. By default the **"/opt"** directory has root privileges which are changed to allow a regular user to access.

```
user@ubuntu:~/$ ls -ld /opt
drwxr-xr-x 2 root root 4096 2013-05-16 10:47 /opt
user@ubuntu:~/$ sudo chmod 777 /opt
[sudo] password for user:
user@ubuntu:~/$ ls -ld /opt
drwxrwxrwx 2 root root 4096 2013-05-16 10:47 /opt
```

2.2. Install the LTIB

The **LTIB** (Linux Target Image Builder) is a tool that can be used to develop and deploy BSPs (Board Support Packages) for a number of embedded target platforms including PowerPC, ARM.

The VAB-820-X Solution Pack is developed based on Freescale released i.MX6x BSP **"L3.0.35_4.1.0_130816_source.tar.gz"**. Users can get it from Freescale official web site.

2.2.1. Extracting Bundle and Installing the LTIB

This section describes the steps to extract the content from the source bundle and to install LTIB.

2.2.1.1. Download i.MX6 Linux Source Bundle

The Linux source bundle can be found and downloaded at:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX6Q&fpsp=1&tab=Design_Tools_Tab

The Run-time Software section in Software & Tools tab is shown in Figure 2.

Figure 2. Source Code download link

Using Firefox, the default save location is in the Downloads folder that contains the file: **"L3.0.35_4.1.0_130816_source.tar.gz".**

2.2.1.2. User install

All LTIB installation and execution should be done by a regular user instead of a root account.

To accommodate running LTIB as a regular user and allow this process to perform privileged commands requiring root permissions that the sudoer's file is modified (see section 2.1.2), and the permissions on the /opt directory are changed (see section 2.1.6).

2.2.1.3. Extract the content

We assume that the file **"L3.0.35_4.1.0_130816_source.tar.gz"** is already in the **"/home/user/imx6/"** folder directory.

```
user@ubuntu:~$ cd ~/imx6/
user@ubuntu:~/imx6/$ ls -l
total 1047352
-rwxr-xr-x 1 user user 896737878 2013-04-09 16:37
L3.0.35_4.1.0_130816_source.tar.gz

user@ubuntu:~/imx6$ tar -zxf L3.0.35_4.1.0_130816_source.tar.gz
user@ubuntu:~/imx6$ cd L3.0.35_4.1.0_130816_source/
user@ubuntu:~/imx6/L3.0.35_4.1.0_130816_source$ ls
EULA  install  ltib.tar.gz  package_manifest.txt  pkgs  redboot_201003.zip
tftp.zip

user@ubuntu:~/imx6/L3.0.35_4.1.0_130816_source$ ./install

You are about to install the LTIB (GNU/Linux Target Image Builder)
Before installing LTIB, you must read and accept the EULA (End User
License Agreement) which will be presented next.

Do you want to continue ? Y|n
Y
- - - - - - - - - - - - -
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/user/imx6/L3.0.35_4.1.0_130816_source/ltib
Where do you want to install LTIB ?
(/home/user/imx6/L3.0.35_4.1.0_130816_source)
/home/user/imx6/
- - - - - - - - - - - - -
Copying packages to ../ltib/pkgs
Installation complete, your ltib installation has been placed in
/home/user/imx6/ltib, to complete the installation:

cd /home/user/imx6/ltib
./ltib

user@ubuntu:~/imx6/L3.0.35_4.1.0_130816_source$
```

3. Building through LTIB

Once the building environment has been configured as described in Chapter 2, the environment and host are now ready to run the LTIB. This chapter will guide through building the BSP via LTIB.

3.1. Getting iMX6x Based Board Packages

To get iMX6 based board packages through LTIB when you first use the LTIB. Those packages include iMX6 hardware definitions, drivers and many more.

```
user@ubuntu:~/imx6/ltib$ ./ltib -c
```

Depending on the performance of user's computer, this process will take time to run. It may take several minutes to an hour. Once the LTIB environment is configured, a menu will be available for selecting configurations.

The first menu is shown in Figure 3. Use the arrow keys to move the cursor between **<Select>** and **<Exit>.**

Choose **<Select>** then press enter key to open the selected item.

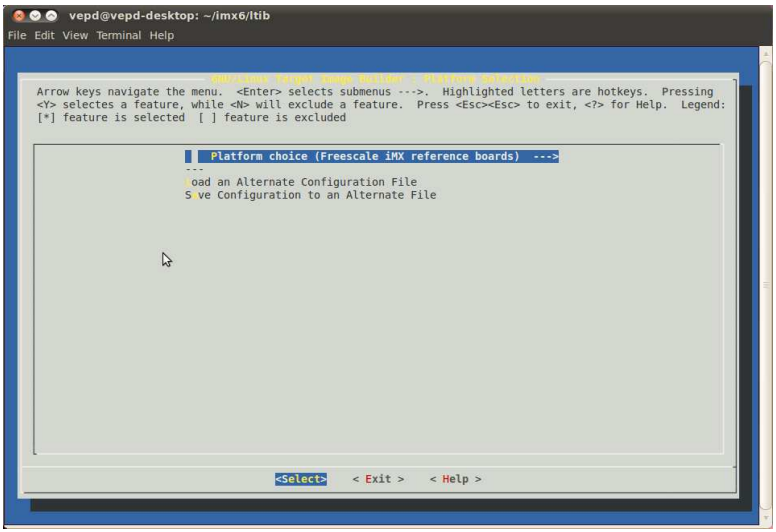


Figure 3. Target Image Builder Platform Selection

In Figure 4, it shows that **<Yes>** has been selected. Press enter key to save.

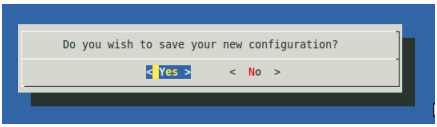


Figure 4. Save Platform Image Selection

The next menu is shown in Figure 5 and provides the target platform to be selected. Using the cursor arrow down key, move the cursor over the **Selection (imx25_3stack)** → and press enter key. The submenu of the selected platform is shown in Figure 6.

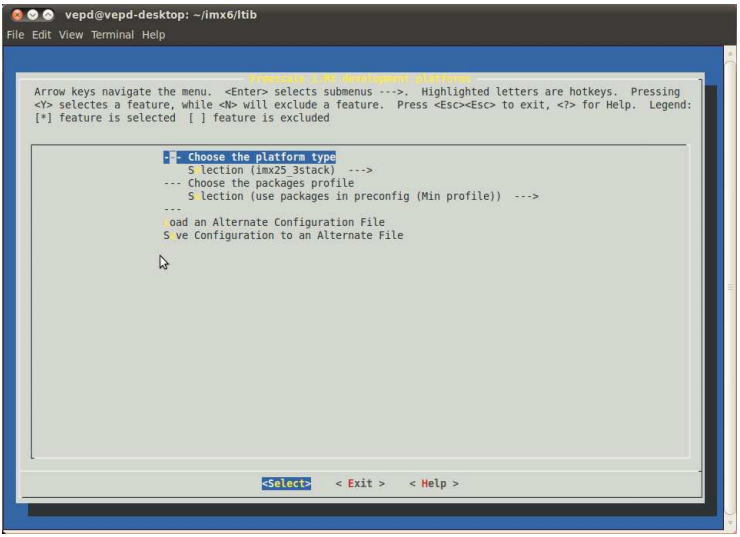


Figure 5. i.MX Development Platforms

In submenu selection, select the **imx6q** platform by moving the cursor over the imx6q as shown in Figure 6, then press enter key.

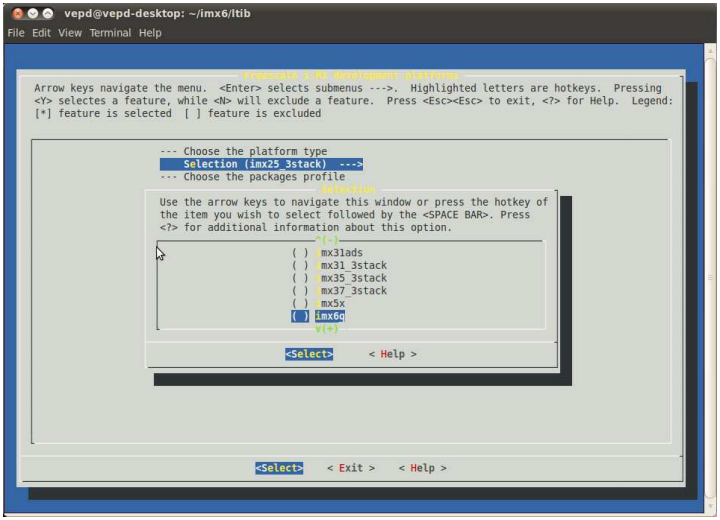


Figure 6. imx6q Platform Selection

Move the cursor to **<Exit>** then press enter key. The save screen is presented as shown in Figure 7. Select **<Yes>** and press enter key to go to the option.

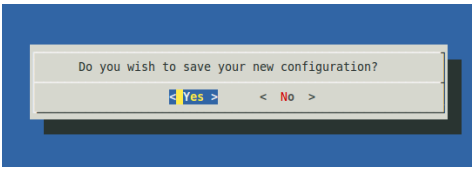


Figure 7. Platform Save

The imx6q Based Boards menu is now presented as shown in Figure 8. The u-Boot board selection must be changed to mx6q_sabrelite. Move the cursor over **board (mx6q_arm2)** and press enter key.

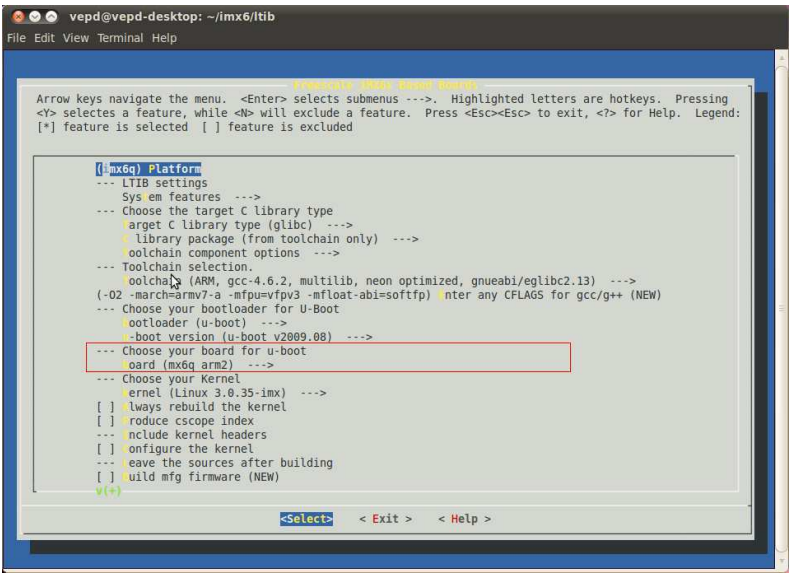


Figure 8. iMX6q Based Boards

Then choose the **mx6q_sabrelite** as shown in Figure 9.

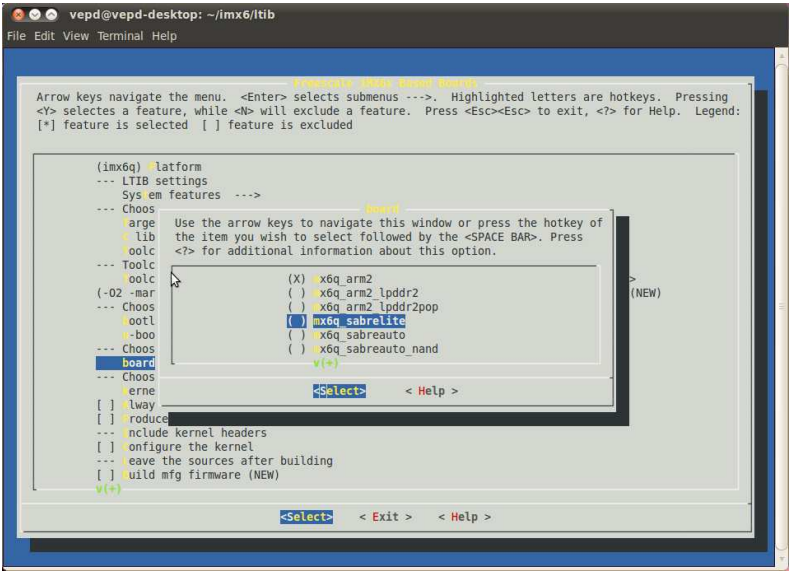


Figure 9. iMX6q_sabrelite

Move the cursor over **<Exit>** and press enter key.

In Figure 10, it shows that **<Yes>** has been selected. Press enter key to save the configuration.

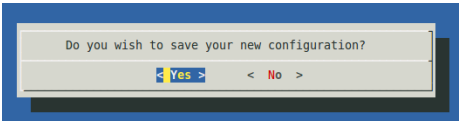


Figure 10. Save the configuration

The sudoer's password is asked for the current user. Enter the password to begin the building process. The building process will take 1.5 hours to complete.

3.2. Building VAB-820 Solution Pack

Once the iMX6 basic packages have been obtained in your building platform as described in section 3.1, this section will guide you through adding or replacing the modification source files as well as building Solution Pack in order to make u-boot, kernel workable on VAB-820.

3.2.1. Add VAB-820 patches to LTIB

Since there are several H/W definitions that are different from original iMX6 source files. User has to add VAB-820 patches in the path below, in order to enable the VAB-820 I/O functions. The LTIB path for this example is **"/home/user/imx6/ltib"**, and the kernel source folder is **"rpm/BUILD/linux/"** under the LTIB path. You can find two patch files vab820-uboot.patch and vab820-kernel.patch at **"BSP/PatchFiles/"** folders.

Step 1

Open **Terminal** utility.

Step 2

Copy **"BSP/PatchFiles/vab820-kernel.patch"** to **"ltib/rpm/BUILD/linux/"**.

```
user@ubuntu:~/$ cd BSP/PatchFiles/
user@ubuntu:~/BSP/PatchFiles$ cp vab820-kernel.patch
 /home/user/imx6/ltib/rpm/BUILD/linux/
```

Step 3

Add **"vab820-kernel.patch"** to kernel source codes.

```
user@ubuntu:~/$ cd /home/user/imx6/ltib/rpm/BUILD/linux/
user@ubuntu:~/imx6/ltib/rpm/BUILD/linux$ patch -p1 < vab820-kernel.patch
```

Step 4

Since LTIB will automatically remove all source files from **"ltib/rpm/BUILD/"** when it finishes building, except the **"linux/"** folder. You need to extract u-boot package manually if you want to modify u-boot files. Run the following command to extract u-boot

```
user@ubuntu:~/$ cd /home/user/imx6/ltib/
user@ubuntu:~/imx6/ltib$ ./ltib -m prep -p u-boot
user@ubuntu:~/imx6/ltib$ ls rpm/BUILD/
linux linux-3.0.35 u-boot-2009.08
```

Then you will find "**u-boot-2009.08/"** folder under **"ltib/rpm/BUILD/"**.

Step 5

Copy **"BSP/PatchFiles/vab820-uboot.patch"** to **"ltib/rpm/BUILD/u-boot-2009.08/"**.

```
user@ubuntu:~/$ cd BSP/PatchFiles/
user@ubuntu:~/BSP/PatchFiles$ cp vab820-uboot.patch
 /home/user/imx6/ltib/rpm/BUILD/u-boot-2009.08/
```

Step 6

Add **"vab820-uboot.patch"** to u-boot source code.

```
user@ubuntu:~/$ cd /home/user/imx6/ltib/rpm/BUILD/u-boot-2009.08/
user@ubuntu:~/imx6/ltib/rpm/BUILD/u-boot-2009.08 $ patch -p1 < vab820-
 uboot.patch
```

3.2.2. Run LTIB to build

The following steps will guide you through building image after adding/replacing the VAB-820 modification files.

Step 1

Type **$./ltib -c**

```
user@ubuntu:~/$ cd /home/user/imx6/ltib/
user@ubuntu:~/imx6/ltib$ ./ltib -c
```

Step 2

LTIB menu will be shown on the screen.

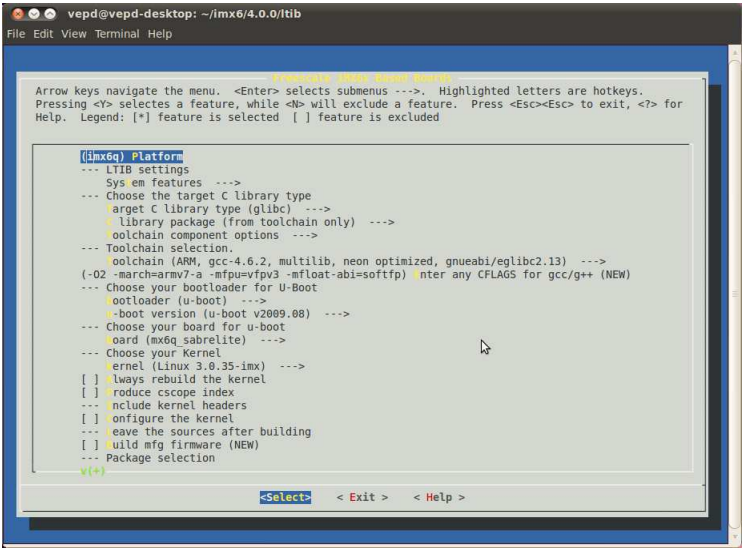


Figure 11. iMX6 Based Boards

Step 3

Select **"Configure the kernel"**.

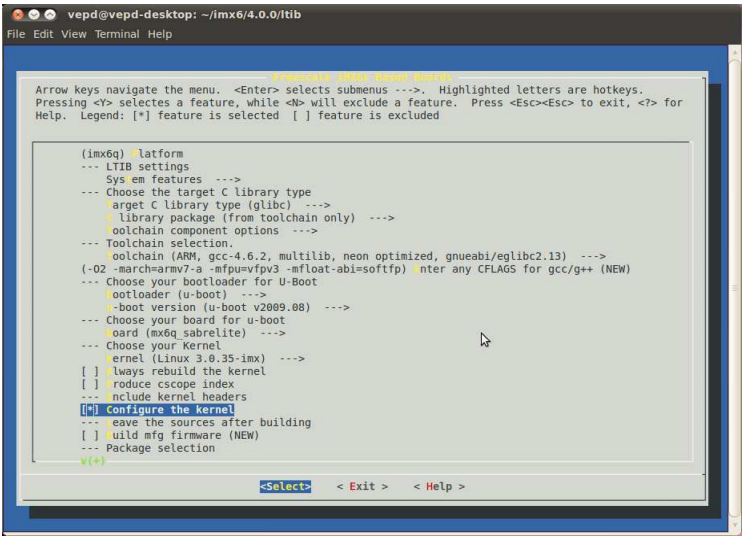


Figure 12. Configure the kernel

Step 4

Move the cursor over **<Exit>** and press enter key.

Step 5

Select **<Yes>** to save the configuration.

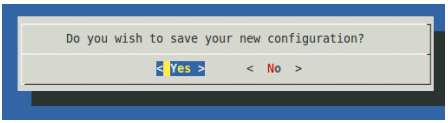


Figure 13. Save the configuration

Step 6

Then the kernel configuration menu will be shown on the screen.

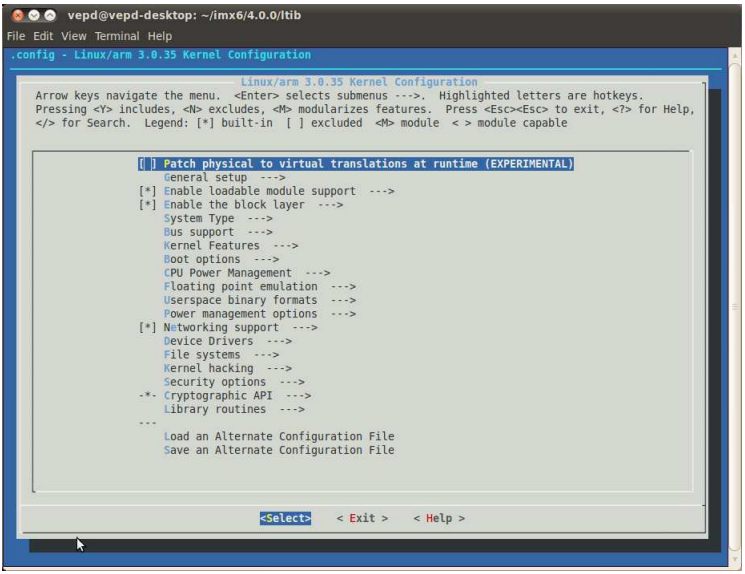


Figure 14. Kernel configuration menu

Step 7

It is recommended to select at least the following options.

```
File systems --> [*] FUSE (Filesystem in Userspace) support
File systems --> DOS/FAT/NT Filesystems -->
        <*> NTFS file system support
        <*> NTFS write support

System type --> Freescale MXC implementations --> [*]PCI Express
support

/* support VNT9485 MiniPCIe module */
Networking support --> Wireless --> <*> Generic IEEE 802.11
Networking Stack
Device Drivers --> Network device support --> Wireless LAN -->
Atheros Wireless Cards -->
        <M> Atheros 802.11n wireless cards support
        [*] Atheros ath9k PCI/PCIe bus support

/* support EMIO-1533 USB Wireless module */
Device Drivers --> Network device support --> Wireless LAN -->
Atheros Wireless Cards -->
        <M> Atheros HTC based wireless cards support

/* support EMIO-2550 3G/GPS module */
Bus support --> <*> PCCard (PCMCIA/CardBus) support
Device Drivers --> USB support --> <*> USB Serial Converter support
-->
        [*] USB Generic Serial Driver
        <*> USB driver for GSM and CDMA modems
Device Drivers --> USB support --> <*> USB Modem (CDC ACM) support

Device Drivers --> Network device support --> <*> PPP (point-to-
point protocol) support -->
        [*] PPP multilink support
        <*> PPP support for async serial ports
        <*> PPP support for sync tty ports
        <*> PPP Deflate compression
        <*> PPP BSD-Compress compression

/* Support HID multi-touch panel */
Device Drivers --> HID Devices --> Special HID drivers -->
        <M> HID Multitouch panel
```

Step 8

It is recommended to remove the following options.

```
Device Drivers --> Input device support --> [*] Keyboards --> [ ]
GPIO Buttons
CPU Power Management --> CPU Frequency scaling --> [ ]CPU Frequency
scaling
```

Step 9

Move the cursor over **<Exit>** and press enter key.

Step 10

Select **<Yes>** to save new kernel configuration.

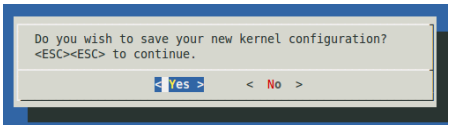


Figure 15. Save the configuration

The sudoer's password is asked for the current user. Enter the password to begin the building process. The building process will take 1.5 hours to complete.

If the building process is successful, you can see the message on the screen.

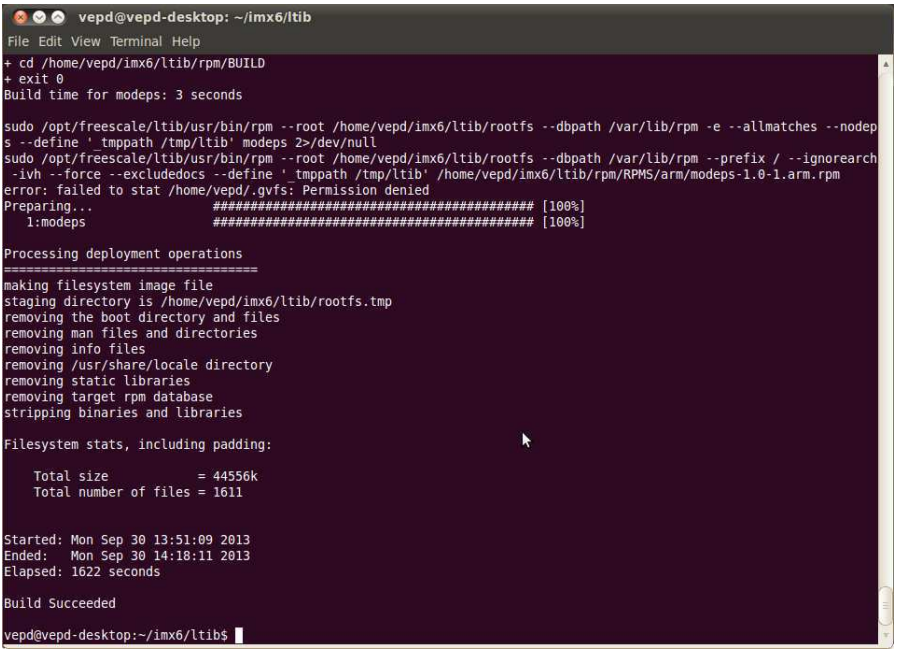


Figure 16. Build successful

There is u-boot.bin, uImage and root file system generated by LTIB. The location for this example can be found in the directory **"/home/user/imx6/ltib/rootfs/"** as shown in Table 1.

Binary	Path	Description
u-boot.bin	~/imx6/ltib/rootfs/boot	U-Boot boot loader
uImage	~/imx6/ltib/rootfs/boot	Kernel
rootfs	~/imx6/ltib/rootfs	Root file system: A folder which includes drivers, library, instruction, and configure files. All you have done in LTIB will be put here.

Table 1. Images generated through LTIB

4. Making Linux System Booting Media

VAB-820 supports two booting ways. One is from Micro SD storage card and the other is SPI ROM. This section will guide you through making the Linux system boot media for VAB-820.

4.1. Making a Linux System Micro SD Storage Card

When you get u-boot.bin, uImage and root file system from LTIB, you can refer to the following sections to make it booting.

4.1.1. Requirements

- Your computer
- Micro SD storage card. Recommended size is 8GB or at least 4GB Class 4.
- SD card reader.

4.1.2. Partition Micro SD storage card

The Micro SD storage card can be identified and auto mounted once inserted to the computer. You can check Micro SD card code name by **$ df -h**

```
user@ubuntu:~/$ df -h
...
Filesystem        Size   Used   Avail   Use%   Mounted on

/dev/sdb1         7.2G   531M   6.3G    8%      /media/usb
```

However, there are some instances that the Micro SD storage card could not identify or auto mount after inserting to the computer. In that case, you can try the other way to identify the Micro SD card.

```
user@ubuntu:~/$ dmesg | grep -i removable
[105.502517] sd 9:0:0:0 [sdb] Attached SCSI removable disk
```

The Micro SD storage card code name for this example is identified as **sdb**. Umount Micro SD storage card before you partition it.

```
user@ubuntu:~/$ sudo umount /dev/sdb1
```

The following steps describe how to partition the Micro SD storage card.

```
user@ubuntu:~/$ sudo fdisk /dev/sdb
Type the following parameters (each followed by <ENTER>):
u          [switch the unit to sectors instead of cylinders]
d          [repeat this until no partition is reported by the 'p'
            command]
n          [create a new partition]
p          [create a primary partition]
1          [the first partition]
16384      [the starting at the offset sector for this example is #16384,
           the size is 8MB, which leaves enough space for the kernel,
           the boot loader and its configuration data. User had to
           create the starting depend on the space for kernel, boot
           loader]
<enter>    [using the default value will create a partition that spans
           to the last sector of the medium]
w         [write the partition table]
```

Note:

Users have to create the partitions which leave enough space for the kernel, the boot loader and its configuration data made by users themselves.

Here, a new partition has been created on Micro SD storage. You have to apply the new partition table immediately, in order to format it.

```
user@ubuntu:~/$ sudo partprobe
```

The file system format for this example is **ext3**, you can type the command to format the partition:

```
user@ubuntu:~/$ sudo mkfs.ext3 /dev/sdb1
```

4.1.3. Copy images to Micro SD storage card

Step 1

Copy u-boot **"u-boot.bin"** to Micro SD storage card.

```
user@ubuntu:~/imx6/ltib/rootfs/boot$ sudo dd if=u-boot.bin of=/dev/sdb
bs=512 seek=2 skip=2
```

The previous u-boot parameters will be stored in SPI ROM, which is identified as **"/dev/mtdblock0"**. If you want to clear the u-boot parameters to default, use the following command:

```
user@ubuntu:~/$ sudo dd if=/dev/zero of=/dev/mtdblock0 bs=512
seek=1536 count=16
```

Step 2

Copy root file system to Micro SD storage card.

You can build your own root file system from LTIB.

The root file system for this example is located at **"/home/user/imx6/ltib/rootfs"**. A folder includes driver modules, Linux instructions and configurations which depend on user's selection in LTIB. User can make it as a compression file (e.g. tar.gz or tar.bz2) or just copy all the files from **"/home/user/imx6/ltib/rootfs"** into Micro SD storage card.

```
user@ubuntu:~/imx6/ltib/rootfs$ sudo tar -cjf rootfs.tar.bz2 *
```

Mount SD card as a folder and decompress the **"rootfs.tar.bz2"** that you made to Micro SD storage card:

```
user@ubuntu:~/$ sudo mkdir /mnt/mountpoint
user@ubuntu:~/$ sudo mount /dev/sdb1 /mnt/mountpoint
user@ubuntu:~/$ cd /mnt/mountpoint
user@ubuntu:/mnt/mountpoint$ sudo tar jxvf rootfs.tar.bz2 ./
```

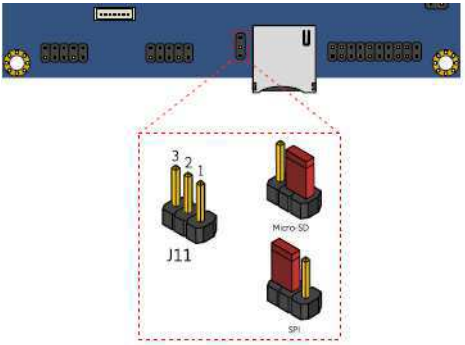
Step 3

Copy the kernel **"uImage"** to Micro SD storage card. The uImage file should be renamed as **"uImage.vab820"** according to u-boot's setting.

```
user@ubuntu:~/imx6/ltib/rootfs/boot$ sudo cp uImage
/mnt/mountpoint/boot/uImage.vab820
```

4.1.4. Setup u-boot parameters for Micro SD card

Setup the u-boot parameter at the first time we boot from Micro SD card. Set the J11 jumper setting (refer to Table 2) to make it boot from Micro SD.



	J11: Boot Select	
	1-2	2-3
★Micro-SD (default)	short	open
SPI	open	short

Table 2. J11 boot selection jumper setting

Connect the VAB-820 and host PC through J5 (COM2) of VAB-820. Run "putty" on host PC to receive the booting message. Power on the VAB-820 and press any key to stop the booting process as shown in Figure 17.

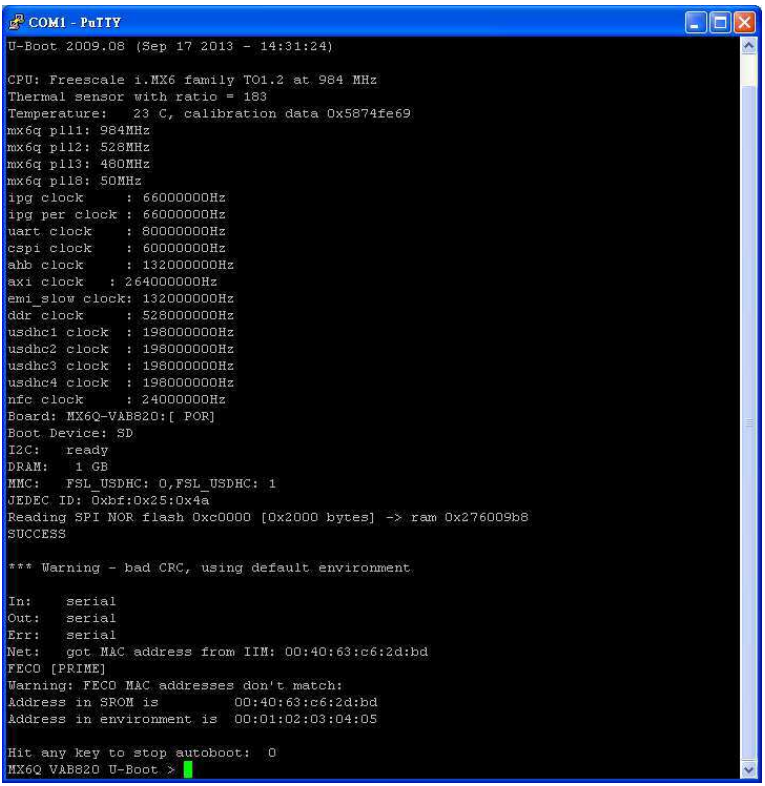


Figure 17. u-boot parameter

To check the parameter in u-boot:

```
VAB-820 U-Boot > pri
bootcmd=run bootcmd_mmc
...
```

The default parameter shows that it loads kernel from eMMC ("bootcmd=run bootcmd_mmc"). You have to set the parameters like the example below. Then the VAB will load kernel from Micro SD card.

```
VAB-820 U-Boot > setenv bootcmd 'run bootcmd_sd'
VAB-820 U-Boot > saveenv
VAB-820 U-Boot > boot
```

4.2. Making a Linux System eMMC

VAB-820 does not support booting from eMMC by default. If you want to make a Linux system on eMMC, there is only one choice to put u-boot on SPI ROM, and put kernel and rootfs on eMMC.

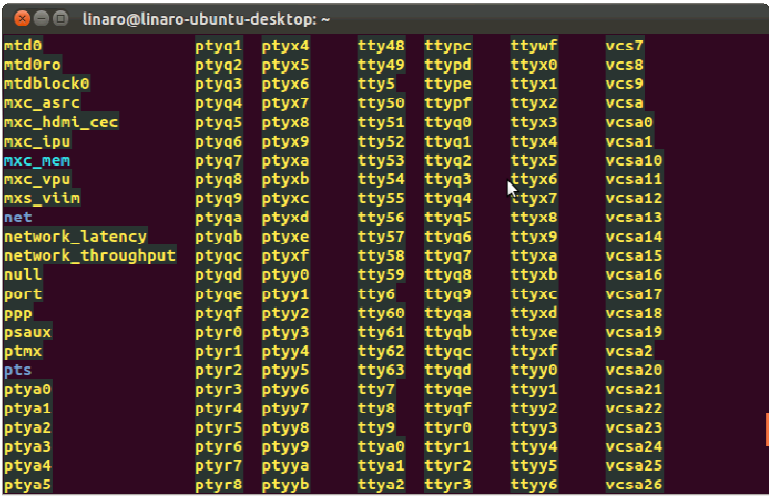
4.2.1. Requirements

- A Linux System Micro SD storage card made in section 4.1 or Appendix A.
- A mass storage includes: **"u-boot.bin"**, **"uImage"** and **"rootfs.tar.bz2"**. You can put those files in a USB pen or SD storage card.

To copy images to SPI ROM and eMMC, you must first boot from Micro SD card on VAB-820.

4.2.2. Burn u-boot.bin into SPI ROM

Run "ls /dev" to check the SPI ROM device, which is identified as **"/dev/mtdblock0"**.



Run the following command to burn **u-boot.bin** into SPI ROM.

```
user@ubuntu:~/$ sudo dd if=u-boot.bin of=/dev/mtdblock0 bs=512
seek=2 skip=2
```

The previous u-boot parameters will be stored in SPI ROM. If you want to clear the u-boot parameters to default, use the following command:

```
user@ubuntu:~/$ sudo dd if=/dev/zero of=/dev/mtdblock0 bs=512
seek=1536 count=16
```

4.2.3. Partition eMMC

eMMC can be identified when booting into VAB-820 from Micro SD card.

```
user@ubuntu:~/$ ls -l | grep -i mmcblk
...
```

Sometimes the eMMC will auto mount if it is the first time of using it. Umount the eMMC before partition it. The eMMC code name for this example is identified as **mmcblk0**.

```
user@ubuntu:~/$ sudo umount /dev/mmcblk0*
```

The following steps on how to partition the eMMC.

```
user@ubuntu:~/$ sudo fdisk /dev/mmcblk0
Type the following parameters (each followed by <ENTER>):
u         [switch the unit to sectors instead of cylinders]
d         [repeat this until no partition is reported by the 'p'
command]
n         [create a new partition]
p         [create a primary partition]
1         [the first partition]
16384     [the starting at the offset sector for this example is #16384,
           the size is 8MB, which leaves enough space for the boot
           loader and its configuration data]
<enter>   [using the default value will create a partition that spans
           to the last sector of the medium]
w         [write the partition table]
```

Note:

Users have to create the partitions which leave enough space for the kernel, the boot loader and its configuration data made by users themselves.

Here is an example; a new partition has been created on eMMC. You have to apply the new partition table immediately, in order to format it.

```
user@ubuntu:~/$ sudo partprobe
```

Make sure to umount the eMMC before doing the steps below.

You can type the command to format the partitions:

```
user@ubuntu:~/$ sudo mkfs.ext3 /dev/mmcblk0p1
```

4.2.4. Copy images to eMMC

User can put uImage/root file system to USB pen or Micro SD card. The storage for this example is an USB pen, and assumes the mount point is /media/usbpen.

Step 1

Copy root file system to eMMC.

The root file system for this example is generated by LTIB. The location is **"/home/user/imx6/ltib/rootfs"**. A folder includes driver modules, Linux instructions and configurations which depend on user's selection in LTIB.

User can make it as a compression file (e.g. tar.gz or tar.bz2). The compression file for this example is rootfs.tar.bz2.

```
user@ubuntu:~/imx6/ltib/rootfs$ sudo tar -cjf rootfs.tar.bz2 *
```

The compression file path for this example is /media/usbpen/.

```
user@ubuntu:~/$ sudo mount /dev/mmcblk0p1 /mnt/mountpoint
user@ubuntu:~/$ cd /mnt/mountpoint
user@ubuntu:/mnt/mountpoint$ sudo tar jxvf
/media/usbpen/rootfs.tar.bz2 ./
...
user@ubuntu:/mnt/mountpoint$ sudo sync && sync
user@ubuntu:/mnt/mountpoint$ cd ~
```

Step 2

Copy kernel **"uImage"** to eMMC.

The image path for this example is /media/usbpen/.

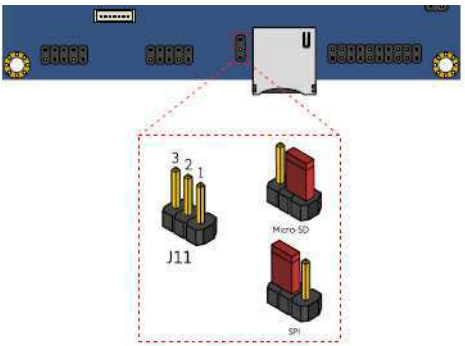
```
user@ubuntu:/media/usbpen /$ sudo cp uImage
/mnt/mountpoint/boot/uImage.vab820
user@ubuntu:/media/usbpen/$ sudo umount /mnt/mountpoint
```

Notes:

1. VAB-820-X Linux BSP won't provide Ubuntu root file system for evaluation actively. Users can get an Ubuntu demo image from Freescale official web site and follow up Freescale's policy to evaluate.
   For more details, refer to Appendix A.
2. For the details on how to make a compression root file system for evaluation, refer to Appendix A.

4.2.5. Setup u-boot parameters for SPI ROM

Setup the u-boot parameter at the first time we boot from SPI ROM. Set the J11 jumper setting (refer to Table 3) to make it boot from SPI ROM.



	J11: Boot Select	
	1-2	2-3
★Micro-SD (default)	short	open
SPI	open	short

Table 3. J11 boot selection jumper setting

Connect the VAB-820 and host PC through J5 (COM2) of VAB-820. Run "putty" on host PC to receive the booting message. Power on the VAB-820 and press any key to stop the booting process as shown in Figure 18.

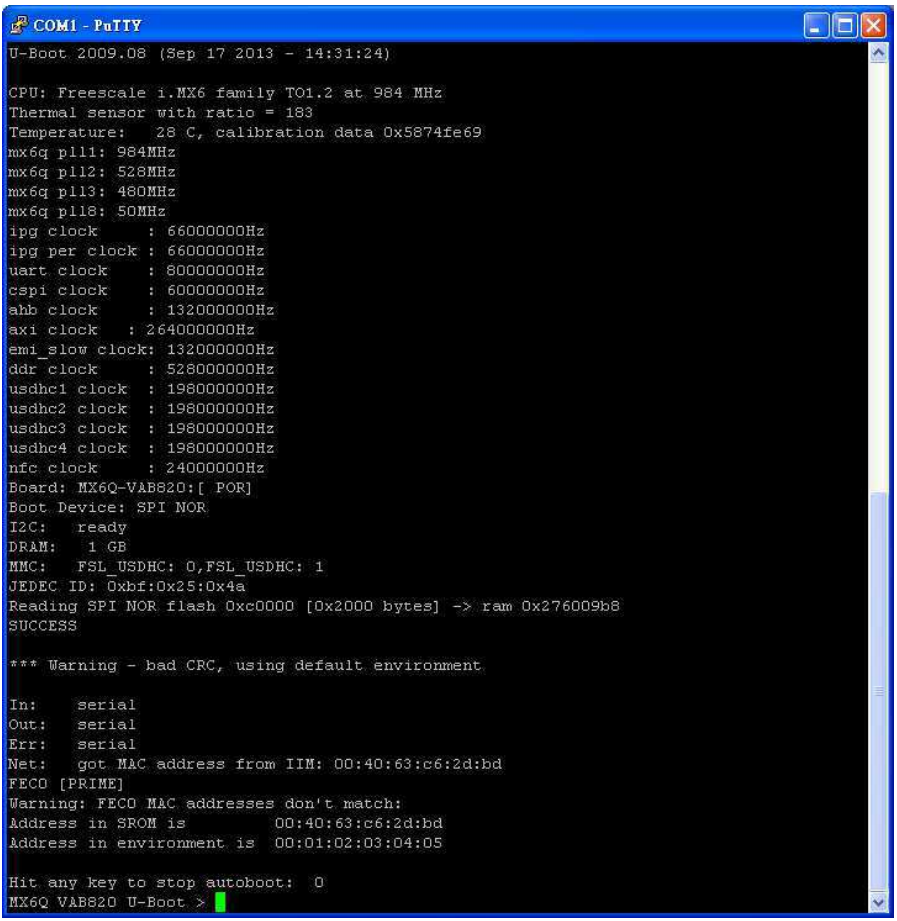


Figure 18. u-boot parameter

To check the parameter in u-boot:

```
VAB-820 U-Boot > pri
bootcmd=run bootcmd_mmc
...
```

The default parameter is "bootcmd=run bootcmd_mmc", which is to load kernel from eMMC. If it is not set like this, you have to set the parameters as the example shown below. Then the VAB will load kernel from eMMC.

```
VAB-820 U-Boot > setenv bootcmd 'run bootcmd_mmc'
VAB-820 U-Boot > saveenv
VAB-820 U-Boot > boot
```

Notes:

User can type "destoryenv" in u-boot to restore the default parameter. For example:
VAB-820 U-Boot > setenv bootcmd 'run bootcmd_mmc'
VAB-820 U-Boot > reset

Appendix A. Making Ubuntu Demo Image

There is a Canonical trademark policy when using Ubuntu in commercial usage or redistribution. The VAB-820-X Linux BSP does not provide Ubuntu DEMO image for evaluation actively. User can follow Freescale's policy and get the demo image from Freescale official web site, if user would like to evaluate Ubuntu on VAB-820.

This section will guide you through making an Ubuntu demo image, then copy it into Micro SD storage card or eMMC.

Required files

Ubuntu file system: You can download Ubuntu file system from Freescale official web site. The file name for this example is **oneiric.tgz** :

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX6Q&fpsp=1&tab=Design_Tools_Tab

EVK/vab820_demo_image.tar.gz: We provide some scripts for you to install demo images on Micro SD card and eMMC.

A.1. Making demo image into Micro SD

Step 1

Prepare a Micro SD storage card (at least 4GB size and Class 4), and insert it into your Linux developing PC (Ubuntu 10.04.x x86 at least).

Step 2

Copy the demo image **"EVK/vab-820_demo_image.tar.bz2"** to your developing PC.

Step 3

Open **Terminal** utility.

Step 4

Untar **"vab-820_demo_image.tar.bz2"**.

```
user@ubuntu:~/$ tar jxvf vab-820_demo_image.tar.bz2
```

Step 5

Put the downloaded file system **oneiric.tgz** under **"vab820_demo_image/"**.

```
user@ubuntu:~/$ cp oneiric.tgz vab-820_demo_image/
```

Step 6

Change directory to **"vab-820_demo_image/"**.

```
user@ubuntu:~/$ cd vab-820_demo_image/
user@ubuntu:~/vab-820_demo_image$
```

Step 7

Run **820_create_sd_fs.sh** script.

```
user@ubuntu:~/vab-820_demo_image$ ./820_create_sd_fs.sh /dev/sdb
```

Step 8

Remove the Micro SD card from your developing PC and insert it into VAB-820. Switch the jumper to boot from Micro SD.

Step 9

Modify the u-boot parameter to load kernel from Micro SD card.

```
setenv bootcmd 'run bootcmd_sd'
```

Step 10

After booting to ubuntu, open Terminal utility and run the script on Desktop to update X11 acceleration files.

```
linaro@linaro:~/$ cd Desktop/820_x11_hw_accel/
linaro@linaro:~/Desktop/820_x11_hw_accel$ ./X11-acceleration-setup.sh
```

A.2. Making demo image into eMMC

Step 1

Copy the demo image **"EVK/vab-820_demo_image.tar.bz2"** to your bootable Micro SD card.

```
user@ubuntu:~/$ cp vab-820_demo_image.tar.bz2 /media/sd_820/home/linaro/
```

Step 2

Open **"Termianl"** utility and untar **"vab-820_demo_image.tar.bz2"**.

```
user@ubuntu:~/$ cd /media/sd_820/home/linaro
user@ubuntu:/media/sdcard/home/linaro$ tar jxvf vab-
820_demo_image.tar.bz2
```

Step 3

Put the downloaded file system **oneiric.tgz** under **"vab820_demo_image/"**.

```
user@ubuntu:~/$ cp oneiric.tgz /media/sd_820/home/linaro/vab-
820_demo_image/
```

Step 4

Insert the Micro SD card into VAB-820 and switch the jumper to boot VAB-820 from Micro SD card.

Step 5

Open **"Terminal"** utility.

Step 6

Change directory to **"vab-820_demo_image/"**.

```
linaro@linaro:~/$ cd vab-820_demo_image/
linaro@linaro:~/vab-820_demo_image$
```

Step 7

Run **820_create_emmc_fs.sh** script.

```
linaro@linaro:~/vab-820_demo_image$ ./820_create_emmc_fs.sh
```

Step 8

Remove the Micro SD card from VAB-820. Switch the jumper to boot from SPI ROM and reboot VAB-820.

Step 9

Modify the u-boot parameter to load kernel from eMMC.

```
setenv bootcmd 'run bootcmd_mmc'
```

Step 10

After booting to ubuntu, open **Terminal** utility and run the script on Desktop to update some X11 acceleration related files.

```
linaro@linaro:~/$ cd Desktop/820_x11_hw_accel/
linaro@linaro:~/Desktop/820_x11_hw_accel$ ./X11-acceleration-setup.sh
```

A.3. Setting u-boot parameters

Step 1

Setting the display devices.

[HDMI]

To set HDMI as display output.

```
setenv bootargs_base 'setenv bootargs console=ttymxc1,115200 ${hdmi}
```

```
vmalloc=320M'
```

To set HDMI resolution.

```
setenv hdmi 'video=mxcfb0:dev=hdmi,1920x1080M@60,if=RGB24'
```

[LVDS]

AMOS-820 supports three LVDS types by default since v1.0.10.

- AUO 22" G220SVN01.0 (1680x1050)
- AUO 10.4" G104XVN01.0 (1024x768)
- AUO 7" G070VW01 V0 (800x480)

User can check parameters in u-boot by typing "pri":

```
lvds_auo_g220=video=mxcfb0:dev=ldb,LDB-WSXGA+,if=RGB24 ldb=spl0
lvds_auo_g140=video=mxcfb0:dev=ldb,LDB-XGA,if=RGB24 ldb=sin0
lvds_auo_g007=video=mxcfb0:dev=ldb,480C60,if=RGB24 ldb=sin0
lvds=video=mxcfb0:dev=ldb,LDB-XGA,if=RGB24 ldb=sin0
```

Check the LVDS power selection setting

J3 : LVDS_power select of AUO 22" G220SVN01.0			
IVDD		PVDD	
*1-3	+12V	2-4	+3.3V
3-5	+5V	*4-6	+5v

J3 : LVDS_power select of AUO 10.4" G104XVN01.0			
IVDD		PVDD	
*1-3	+12V	*2-4	+3.3V
3-5	+5V	4-6	+5v

J3 : LVDS_power select of AUO 7" G070VW01 V0			
IVDD		PVDD	
*1-3	+12V	*2-4	+3.3V
3-5	+5V	4-6	+5v

User can set the LVDS type in u-boot, the LVDS for this example is AUO 10.4" G104XVN01.0:

```
setenv lvds ${lvds_auo_g140}
saveenv
```

To set LVDS as display output.

```
setenv bootargs_base 'setenv bootargs console=ttymxc1,115200 ${lvds}
vmalloc=320M'
```

Step 2

Setting storage devices

[eMMC]

```
setenv bootargs_mmc 'set bootargs ${bootargs} root=/dev/mmcblk0p1 rw
rootwait'
setenv bootcmd_mmc 'run bootargs_base bootargs_sd; mmc dev 1; ext2load
mmc 1:1 $loadaddr $vkernel && bootm'
setenv bootcmd 'run bootcmd_mmc'
```

[Micro SD storage card]

```
setenv bootargs_sd 'set bootargs ${bootargs} root=/dev/mmcblk1p1 rw
rootwait'
setenv bootcmd_sd 'run bootargs_base bootargs_sd; mmc dev 0; ext2load
mmc 0:1 $loadaddr $vkernel && bootm'
setenv bootcmd 'run bootcmd_sd'
```

Step 3

Setting MAC address

Two ways to set MAC address:

[Way 1]

Pass MAC address from u-boot parameter; please ensure that "**ethaddr"** is a valid MAC address. User can set a real MAC address according to sticker on Ethernet PHY.

```
setenv ethaddr 'xx:xx:xx:xx:xx:xx'
setenv bootargs_base 'setenv bootargs console=ttymxc1,115200
fec_mac=${ethaddr} ${hdmi} vmalloc=320M'
```

[Way 2]

Here, user can set the MAC address in eFuse. The address is on the Ethernet physical port.

The MAC Address for this example is 11:22:33:44:55:66.

User can write the MAC address:

```
imxotp blow --force 22 0x33445566
imxotp blow --force 23 0x1122
```

User can check the MAC address:

```
imxotp read 22
0x33445566 (Show the address user write)
imxotp read 23
0x1122(Show the address user write)
```

It is a must to reset after you set MAC address and save:

```
reset
```

Note:
Be careful to write MAC address into eFuse. The reason is that the eFuse can only be written once.

A.4. Update apt repository source list

User has to manually update the Ubuntu package repository source due to Canonical moves Ubuntu 11.10 to old release server.

Step 1

Open **Terminal** utility.

Step 2

Copy **"sources.list"** from EVK folder **to** /etc/apt and update it manually.

```
user@ubuntu:~/$ sudo cp <yourpath>/EVK/sources.list /etc/apt
user@ubuntu:~/$ sudo apt-get update
```

Note:
Although most of the Ubuntu packages can be updated successfully, some of them which provided by linaro may fail to update.

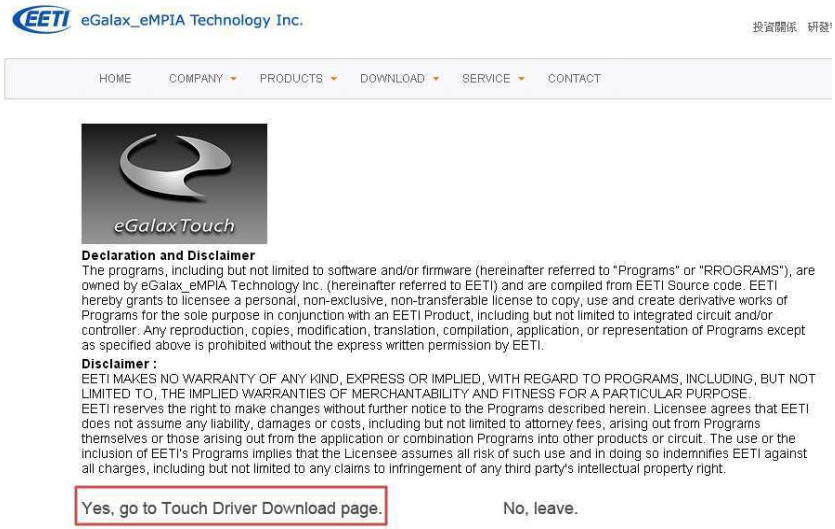
Appendix B. Touch Panel Calibration

The Touch panel type for this example is just for TP220C01 V0(AUO G220SVN01.0) only.

Step 1

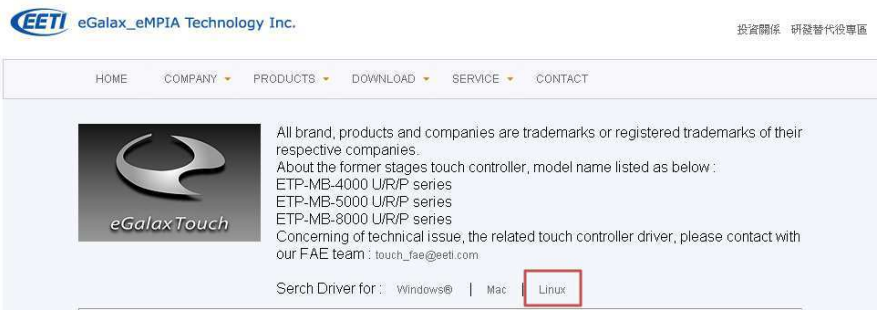
Download Linux Driver package under EETI official web site "**http://home.eeti.com.tw/drivers.html**".

Click **"Yes, go to Touch Driver Download page"** to get driver package.



Step 2

Click "Linux" then down "ARM/MIPS "eGTouch_v2.5.3120.L-ma". The **"eGTouch_v2.5.3120.L-ma.zip"** will be downloaded to user's local storage.



eGalaxTouch Driver for **Linux**

The Linux public driver supports most of the Linux distribution, including Ubuntu, Debian, SuSE(openSuSE), Fedora Core, Mandriva, Slackware
and so on. Please according to your kernel version, download corresponding driver.

Kernel Version	CPU Type	USB / UART / PS2		
Kernel 2.6.24 Upwards	X86 (32/64bits)	2013 / 07 / 20 Download	eGTouch_v2.5.3120.L-x	1. Available for multi-touch as kernel version is 2.6.36 above. If kernel version below 2.6.35, it could only support single point. 2. Available for non-Xwindow system. 3. Support Multi-controller & Multi- monitor. 4. Support Right-Click
	ARM / MIPS	2013 / 07 / 20 Download	eGTouch_v2.5.3120.L-ma	

Step 3

Unzip **"eGTouch_v2.5.3120.L-ma.zip"**. A folder "eGTouch_v2.5.3120.L-ma" will be created.

```
$unzip eGTouch_v2.5.3120.L-ma.zip
```

Step 4

Before running install setup script, please plug-in the controller first. Then you could execute

Execute script file "setup.sh" to install the driver automatically.

```
$sudo sh setup.sh         # To install the eGTouch driver.
$sudo sh setup.sh uninstall  # To remove the eGTouch driver.
```

For more detailed information, user can refer to the guide **"EETI_eGTouch_Linux_Programming_Guide_v2.5f.pdf"** under **eGTouch_v2.5.3120.L-ma\Guide**

Step 5

Execute eCalib to process calibration procedure.

Please execute tools under "root" permission!

```
$sudo eCalib
```

eCalib: The tool eCalib is a calibration tool with command line. Please type "eCalib -h" to see the usage content.

User can select 4 or 9 point to calibrate.

Appendix C. EMIO-2550 GPS function test

Step 1

Enable 3G LAN connection.

Step 2

Install minicom package.

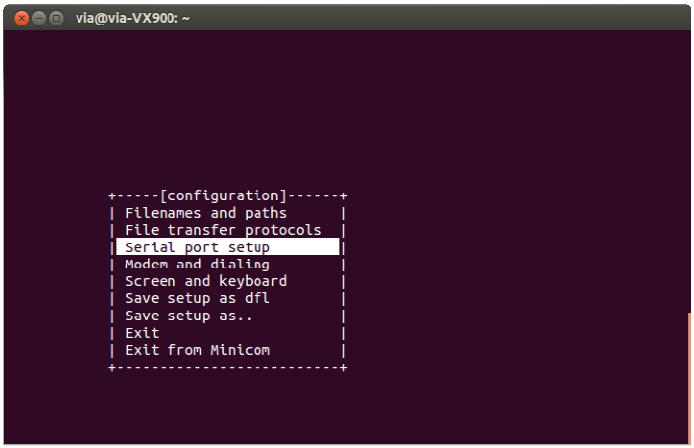
```
$sudo apt-get install minicom
```

Step 3

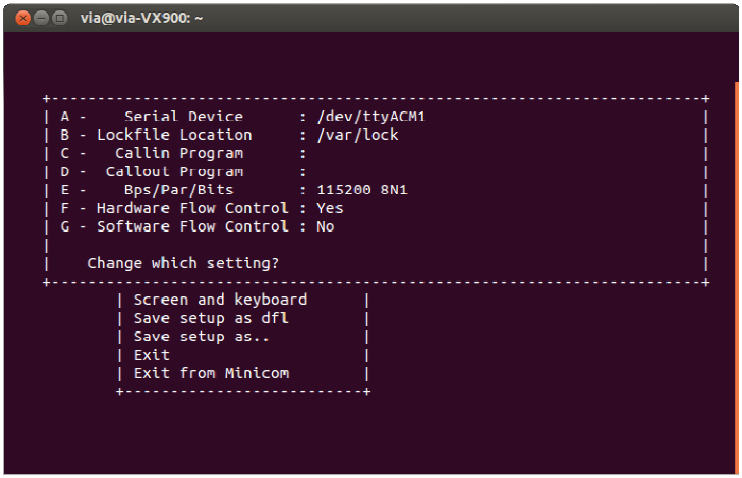
Open /dev/ttyACM1 by minicom

```
$ sudo minicom -s
```

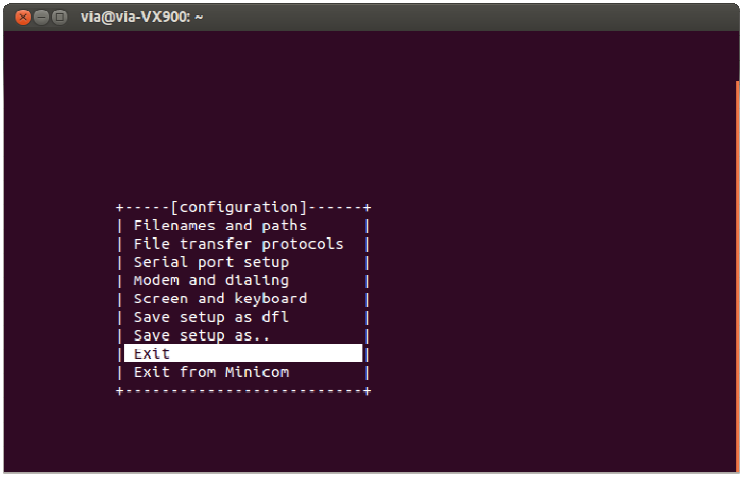
Select "Serial port setup"



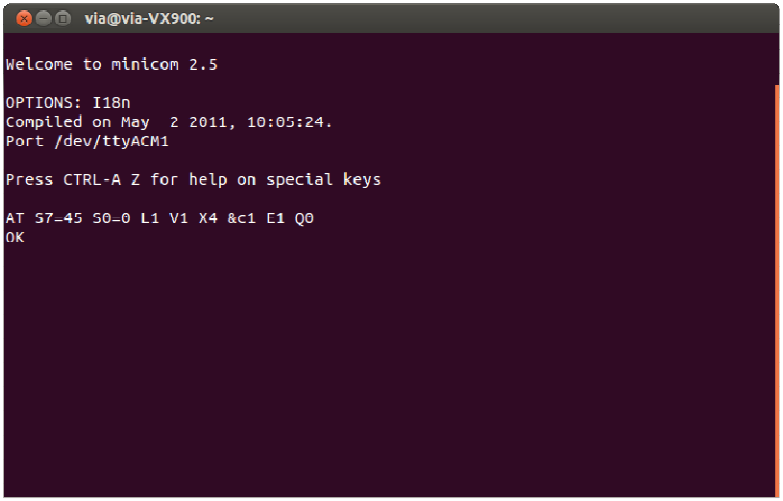
Press “a” key to set the Serial Device as /dev/ttyACM1, and press enter key twice to save and exit.



Select "Exit"



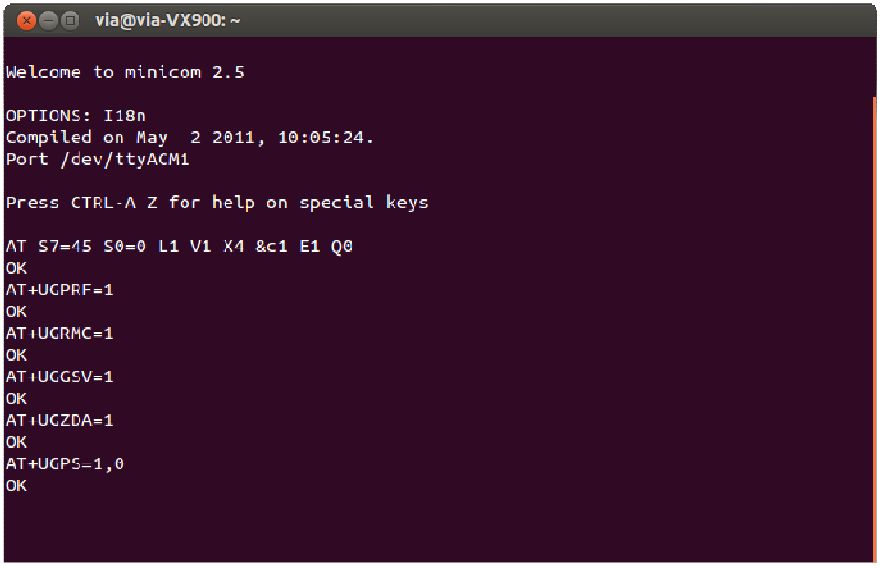
Now the terminal is in minicom tool.



Step 4

Type GPS AT command for testing by minicom tool:

1. **AT+UGPRF=1** (Configure the data flow to and from a u-blox GPS receiver which is connected to the Wireless Module)
2. **AT+UGRMC=1** (get longitude and latitude)
3. **AT+UGGSV=1** (Optional for checking "time" from GPS satellite)
4. **AT+UGZDA=1** (Optional for checking "how many GPS satellite")
5. **AT+UGPS=1,0** (Switches to a u-blox GPS receiver which is connected to the Wireless Module via a dedicated DDC (I2C) interface)

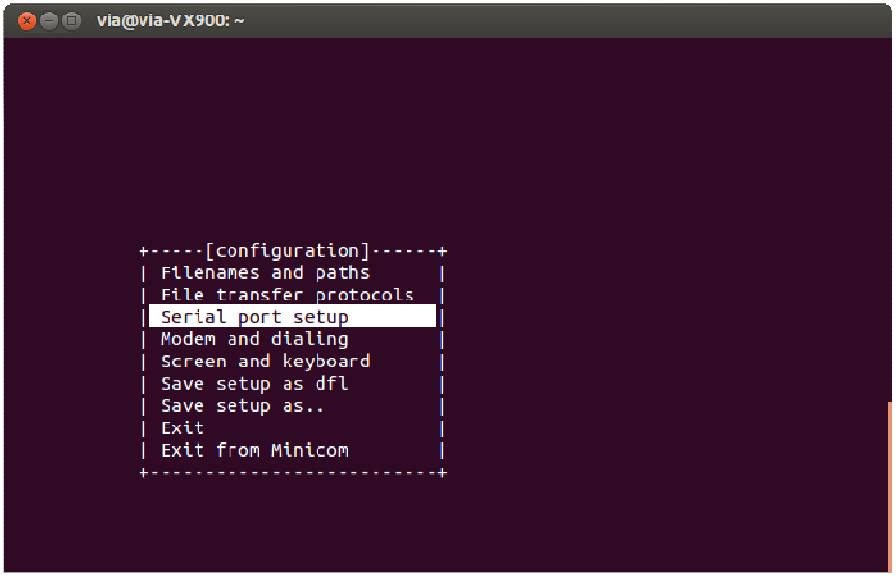


Step 5

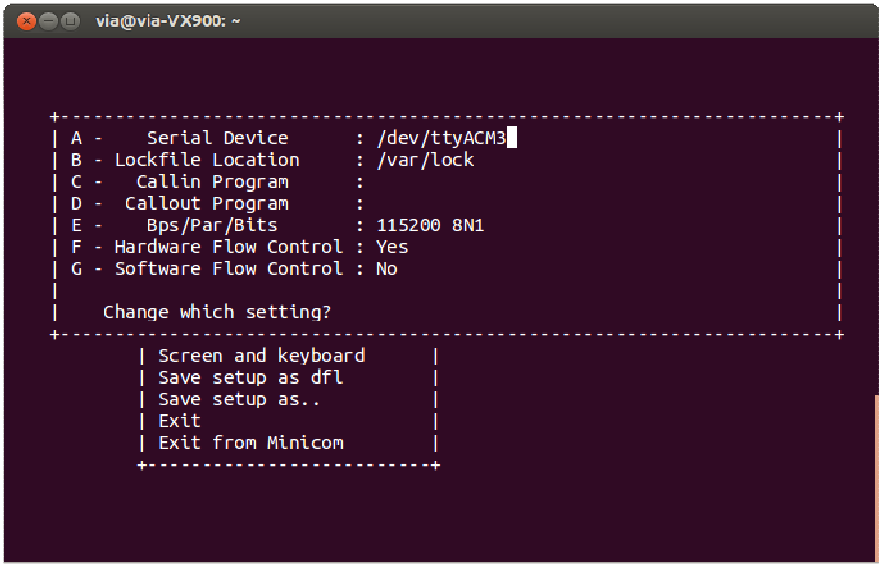
Open another terminal, then open /dev/ttyACM3 by minicom to print GPS receiver data by loop.

```
$ sudo minicom -s
```

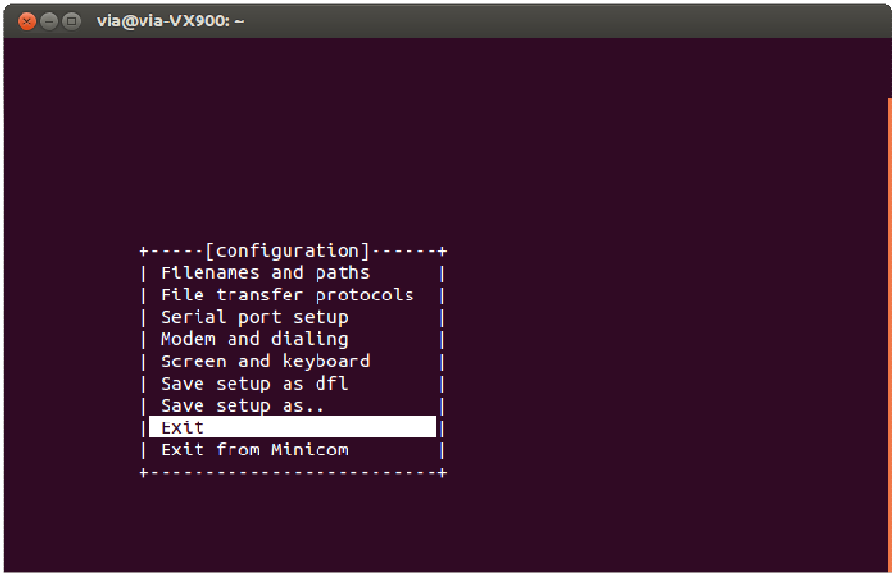
1. Select "Serial port setup"



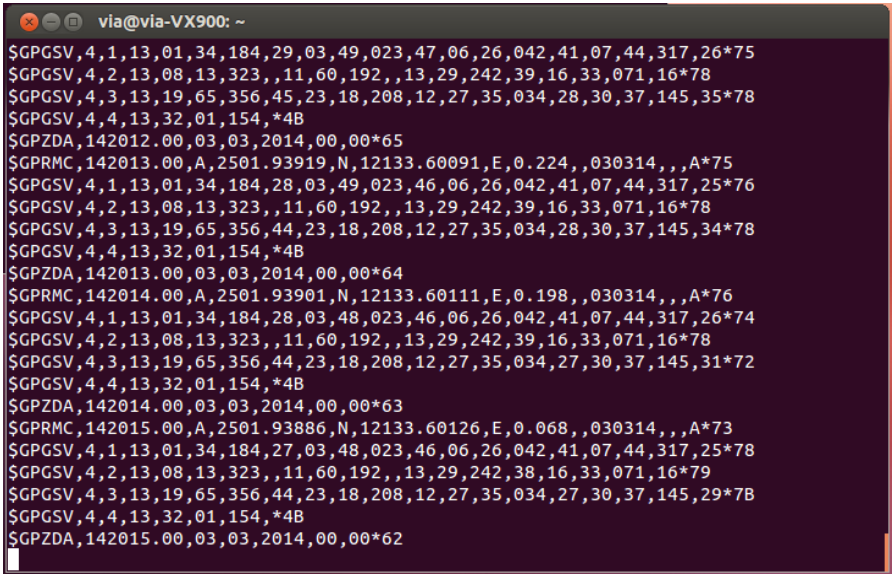
2. Press "a" key to set the Serial Device as /dev/ttyACM3, then press enter key twice to save and exit.



3. Select "Exit"



4. minicom should print the GPS receiver data automatically.



This information is NMEA sentences; please refer to the website to get more information:

- http://www.gpsinformation.org/dale/nmea.htm#nmea
- http://annheilong.pixnet.net/blog/post/24919514-%E3%80%90%E9%9B%BB%E8%85%A6%E3%80%91nmea%E6%A8%99%E6%BA%96%E6%A0%BC%E5%BC%8F

[How to check the fix position]

In the $GPRMC sentence, we can get longitude and latitude of the fix position. For example:

```
$GPRMC,085235.00,A,2459.14178,N,12131.97598,E,1.283,,180713,,,A*76
```

	example	unit
Latitude	2459.14178	DDmm.mmmm
Longitude	12131.97598	DDDmm.mmmm

D: Degree, m: minute

In this example, the fix position is latitude 24 degree 59.14178 minute, longitude 121 degree 31.97598 minute. We can transfer the position to google map to check whether the fix position is correct or not. The unit that Google map can recognize is degree, so we need to transfer the unit first.

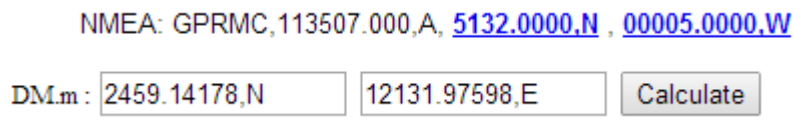
Way 1: Manually

Latitude: 24 deg. 59.14178 min.= 24 deg. + 59.14178/60 deg. = 24.985696 deg.

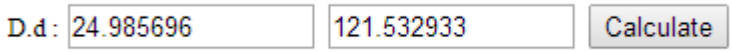
Longitude: 121 deg. 31.97598 min. = 121 deg. +31.97598/60 deg. = 121.532933 deg.

Way 2: Input the position information to this website

- **http://www.hiddenvision.co.uk/ez/**



Press **Calculate** button, it will automatically transfer the unit to degree:



Then we can fill up the latitude and longitude at the following link:
https://maps.google.com/maps?q=---,---

Example: **https://maps.google.com/maps?q=24.985696,121.532933**

Appendix D. EMIO-1533 USB Wireless Module

To enable the EMIO-1533 USB wireless modules, user has to do the followings:

- Select "Atheros HTC based wireless cards support" in kenel menuconfig. (refer to Page 17)
- Put htc_9271.fw to /lib/firmware/. User can download it here: http://wireless.kernel.org/en/users/Drivers/ath9k_htc

Note:
Please make sure ath9k_htc.ko is here: built from ltib to /lib/modules/3.0.35-2666-gbdde708/kernel/drivers/net/wireless/ath/ath9k/

Appendix E. Compile X window driver

Dual display is supported by VAB-820 2GB RAM SKU since AMOS-820 HMI SP v1.0.10. Users can evaluate from EVK directly or follow the steps listed below to build X windows driver if users would like to modify/build by themselves.

Step 1

Copy X Window Driver source package "xserver-xorg-video-imx-viv-3.0.35-4.1.0.tar.bz2" from "BSP/Pkg_External/Graphic/" to Ubuntu evaluation image where runs on VAB-820.

Step 2

Access xserver-xorg-video-imx-viv-3.0.35-4.1.0 folder after untar xserver-xorg-video-imx-viv-3.0.35-4.1.0.tar.bz2.

```
$cd xserver-xorg-video-imx-viv-3.0.35-4.1.0
```

Step 3

Install xserver development package from Ubuntu repositories.

```
$sudo apt-get install xserver-xorg-dev x11proto-xf86dri-dev x11proto-render-dev libdrm-dev
```

Step 4

Compile X Window Driver

```
$./fastbuild.sh
```

Step 5

X window driver will be generated at EXA/src/vivante_drv.so and DRI_1.10.4/src/libdri.so. User can execute the following command to install the drivers.

```
$sudo ./fastbuild.sh install
```

Or copy drivers to the following path manually:

```
$sudo cp EXA/src/vivante_drv.so /usr/lib/xorg/modules/drivers
$sudo cp DRI_1.10.4/src/libdri.so /usr/lib/xorg/modules/extensions
$sudo depmod -a && ldconfig
```

Appendix F. Compile Gstreamer plugin

Users can evaluate Freescale multimedia from EVK directly or follow the steps listed below to build relational plugin if users would like to modify/build by themselves.

Step 1

Copy gstreamer plugin source packages "gst-fsl-plugins-3.0.7.tar.bz2" "gst-pugins-good-0.10.30.tar.bz2" from "BSP/Pkg_External/Multimedia/" to Ubuntu evaluation image where runs on VAB-820.

Step 2

To compile gst-fsl-plugins-3.0.7

```
$tar jxvf gst-fsl-plugins-3.0.7.tar.bz2
$cd gst-fsl-plugins-3.0.7
$./autogen.sh --prefix=/usr PLATFORM=MX6 \
IPU_CFLAGS="-I/usr/include -I/usr/src/linux/include" IPU_LIBS=-lipu \
VPU_CFLAGS="-I/usr/include -I/usr/src/linux/include" VPU_LIBS=-lvpu
$make
$sudo make install
```

Step 3

To compile gst-pugins-good-0.10.30.tar.bz2

```
$tar jxvf gst-pugins-good-0.10.30.tar.bz2
$cd gst-pugins-good-0.10.30
$./configure --prefix=/usr
$make
$sudo make install
```

Taiwan Headquarters
1F, 531 Zhong-Zheng Road
Xindian, Taipei, 23148
Taiwan

TEL: 886.2.2218.5452
FAX: 886.2.2218.5453
Email: embedded@via.com.tw

USA
940 Mission Court
Fremont, CA 94539
USA

TEL: 1.510.683.3300
FAX: 1.510.687.4654
Email: embedded@viatech.com

Europe
In den Dauen 6
53117 Bonn
Germany

TEL: 49.228.688565.0
FAX: 49.228.688565.19
Email: embedded@via-tech.eu

China
Tsinghua Science Park Bldg. 7
No. 1 Zongguancun East Road
Haiden District, Beijing, 100084
China

TEL: 86.10.59852288
FAX: 86.10.59852299
Email: embedded@viatech.com.cn

Japan
3-15-7 Ebisu MT Bldg. 6F
Higashi, Shibuya-ku
Tokyo 150-0011
Japan

TEL: 81.3.5466.1637
FAX: 81.3.5466.1638
Email: embedded@viatech.co.jp

Korea
2F, Sangjin Bldg., 417
Dogok Dong, Gangnam-Gu
Seoul 135-854
South Korea

TEL: 82.2.571.2986
FAX: 82.2.571.2987
Email: embedded@via-korea.com