Tizen IVI Getting Started Guide For PandaBoard

Tizen the True Open Platform

4/30/2013
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>4/30/2013</td>
<td>Added Bluetooth enabling details</td>
</tr>
<tr>
<td>0.2</td>
<td>4/30/2013</td>
<td>Added the initial Review Comments</td>
</tr>
<tr>
<td>0.1</td>
<td>11/26/2012</td>
<td>Initial Draft</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Building wireless drivers</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>WLAN Calibration</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Enabling Bluetooth</td>
<td></td>
</tr>
<tr>
<td>10.1</td>
<td>Kernel Configurations</td>
<td></td>
</tr>
<tr>
<td>10.2</td>
<td>Board Specific Code Modifications</td>
<td></td>
</tr>
<tr>
<td>10.3</td>
<td>Compile the kernel and compile the kernel modules</td>
<td></td>
</tr>
<tr>
<td>10.4</td>
<td>Downloading the Firmware</td>
<td></td>
</tr>
<tr>
<td>10.5</td>
<td>User Space components</td>
<td></td>
</tr>
<tr>
<td>10.6</td>
<td>Testing</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Enabling the sound</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Enabling the Display</td>
<td></td>
</tr>
</tbody>
</table>
1 Panda Board Introduction

The OMAP4430 SoC on the PandaBoard features a dual-core 1 GHz ARM Cortex-A9 MPCore CPU, a 304 MHz PowerVR SGX540 GPU, a C64x DSP, and 1 GiB of DDR2 SDRAM. The PandaBoard ES uses a newer SoC, with a dual-core 1.2 GHz CPU and 384 MHz GPU. Primary persistent storage is via an SD Card slot allowing SDHC cards up to 32 GB to be used. The board includes wired 10/100 Ethernet as well as wireless Ethernet and Bluetooth connectivity. Its size is slightly larger than the ETX/XTX Computer form factor at 4 in x 4.5 in (100 mm x 110 mm). The board can output video signals via DVI and HDMI interfaces. It also has 3.5 mm audio connectors. It has two USB host ports and one USB On-The-Go port, supporting USB 2.0.

2 H/W and S/W requirements

Hardware Requirements: Pandaboard, <=8 GB SD card, HDMI cable, power cable, USB OTG cable, Monitor/TV with hdmi support.

Software Requirements: PC with Ubuntu 11.10 or more, Minicom,

3 Formatting the SD card

For Tizen IVI porting SD card has to be formatted into 4 partitions.

1. **Boot size** > 1GB Filesystem: Vfat
2. **Platform.img** > 3 GB / Filesystem:ext4
3. **Data.img** > 1 GB /opt/ Filesystem:ext4
4. **UMS.img** > 500MB Filesystem: ext4

4 Preparation

Before you start, you need to know the device name for your SD-Card.

4.1. Getting the device name for your SD-Card

If you have no idea which device name is correct run the following command.

```
$ sudo fdisk -ls
```

The device name you are looking for is the Disk which comes close the size of your SD-Card (e.g. for a 8 GB SD-Card one line might look like this: `Disk /dev/sd<X>: 8011 MByte, 801120640 Byte. - so the device name for the SD-Card will be /dev/sde`).

4.2. Step by Step Instructions to format SD card

Since putting a Linux file system on a FAT32 partition is problematic, it is recommended to also create a 2nd partition.

- Insert your SD card into your Linux box
- Do not mount it
The card shows up as /dev/sd*. To identify the card, you can either do:

- `dmesg | grep sd[a-z]`. You will see something like `[172407.246308] sdb: sdb1 sdb2 so drive is /dev/sdb`
- `ls /dev/sd*` before and after plugging SD card and find created devices

For this example

- we will assume the card shows up as /dev/sdc - substitute this for the real device on your specific machine. Fdisk the drive and print the partition information

```
sudo fdisk /dev/sd<X>
Command (m for help): p
Disk /dev/sdc: 1018 MB, 993001472 bytes
...<more>...
```

Look for the size in bytes of the device and calculate the number of cylinders, dropping fractions, if we have 255 heads and 63 sectors (and 512 bytes per sector so 1 cylinder is 255 * 63 * 512 = 8225280 bytes).

```
new_cylinders = Size / 8225280 (for this example we will have 993001472 / 8225280 which equals 120.725 or 120 cylinders)
```

### 4.3. Delete existing Partitions

Since we are changing the underlying geometry of the disk, we must clear the partition table before doing it. So delete all partitions using the fdisk 'd' command - yes, you will lose all data on the card. Once that is done, we can set the new geometry in expert mode. We will set the number of heads to 255, number of sectors to 63, and number of cylinders to new_cylinders.

```
Command (m for help): d
Partition number (1-4): 1
Command (m for help): d
Partition number (1-4): 2
```

### 4.4. Configure SD Card

```
Command (m for help): x
Expert command (m for help): h
Number of heads (1-256, default 30): 255
Expert command (m for help): s
Number of sectors (1-63, default 29): 63
Warning: setting sector offset for DOS compatibility
Expert command (m for help): c
Number of cylinders (1-1048576, default 2286): <new_cylinders calculated from above>
```
4.5. Configure SD Partitions

Now we return to the main menu and create 4 partitions as needed - 1 boot partition of >1GB and the rest a linux partitions.

Expert command (m for help): r
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-123, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-123, default 123): +64M (see note above)
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (10-123, default 10):
Using default value 10
Last cylinder or +size or +sizeM or +sizeK (10-123, default 123):
Using default value 123

4.6. Fat32 Partition

Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))
* You have to format 1st partitions with vfat32 filesystem.
Command (m for help): a
Partition number (1-4): 1

4.7. Check Partition Table

The partition table should look something like the following. Notice the heads, sectors, and cylinders. Make sure partition 1 is active and FAT32. If it looks good - write the new partition information out.

Command (m for help): p
Disk /dev/sdc: 993 MB, 993001472 bytes
255 heads, 63 sectors/track, 120 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000

Device Boot Start End Blocks Id System
/dev/sd<X>1  *  1 9 72261 c W95 FAT32 (LBA)
/dev/sd<X>2 10 120 891607+ 83 Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x partitions, please see the fdisk manual page for additional information.
Syncing disks.

4.8. Formatting Partitions

Format the filesystems on the partitions:

```
# sudo mkfs.vfat -F 32 -n boot /dev/sd<X>1
# sudo mkfs.ext4 -L rootfs /dev/sd<X>2
# sudo mkfs.ext4 -L data /dev/sd<X>3
# sudo mkfs.ext4 -L UMS /dev/sd<X>4
```

4.9. Creating a mount point

```
mkdir /tmp/mmc1
mkdir /tmp/mmc2
mkdir /tmp/mmc3
mkdir /tmp/mmc4

sudo mount /dev/sd<X>1 /tmp/mmc1
sudo mount /dev/sd<X>2 /tmp/mmc2
sudo mount /dev/sd<X>3 /tmp/mmc3
sudo mount /dev/sd<X>4 /tmp/mmc4
```
4.10. Problem seen with FDISK not erasing the first sector

The fdisk utility does not seem to erase the first few bytes of the first sector in the card when the partition table is saved.

Use dd to erase the first sector.

```bash
sudo dd if=/dev/<image-name> of=/dev/<sdc> bs=1024 count=1
```

Then use the procedure listed in section above to create new partitions and format them accordingly.

5 Mounting the Images

The first partition will be the boot partition and copy the X-loader, uboot and Kernel Image (uImage) into it.

To the Tizen images ex. Platform.img, data.img and UMS.img use the following commands.

```bash
sudo dd if=<input file> of=/dev/sd<X> bs=4096 conv=notrunc
```

6 Building the Bootloaders

The following steps will build the x-loader and u-boot.

Note: In order to build the x-booter or u-boot, it is recommended that you use the omappedia Release Notes page corresponding to the platform you are using (Blaze, Blaze Tablet, Panda, or Zoom) and follow the instructions for that release: [http://www.omappedia.com/wiki/Release_Notes](http://www.omappedia.com/wiki/Release_Notes). In general, the instructions will be very similar with the following exceptions:

- The commit ID used to pull the code from the u-boot and x-loader git trees varies based on the release. It is recommended that you use the correct commit ID corresponding to your release, as this is how the validation is performed.
- The config file used to setup the configuration when building the u-boot and x-loader varies based on the platform you are using. See the tables below.

Also, ensure that you have installed the pre-requisite packages for building the bootloaders, namely Git and the CodeSourcery ARM Compiler. These instructions are included in the omappedia Release Notes corresponding to your release. You can also find the information here:

- Git is available for download at [git-scm.com](http://git-scm.com). For more details on installing and using git please see [this wiki page](http://this wiki page).
- For the CodeSourcery ARM Compiler, visit [Support Tools/Cross Compilers](http://Support Tools/Cross Compilers) for further information.
6.1 u-boot

6.1.1 Downloading u-boot source

Clone the u-boot git tree and checkout the source code corresponding to this release:

```
# mkdir u-boot
# cd u-boot
# git clone git://git.omapzoom.org/repo/u-boot.git
# git checkout <<commit_ID_for_uboot_in_this_release>>
```

6.1.2 Building u-boot

Make the u-boot source code using the correct config file depending on your platform:

```
# cd u-boot
# make distclean
# make CROSS_COMPILE=arm-none-linux-gnueabi-
# <<config_file_depending_on_your_platform>>
# make CROSS_COMPILE=arm-none-linux-gnueabi-
```

6.1.3 u-boot config files

<table>
<thead>
<tr>
<th>Board</th>
<th>config file</th>
</tr>
</thead>
<tbody>
<tr>
<td>44xx Blaze Tablet</td>
<td>omap44XXtablet_config</td>
</tr>
<tr>
<td>44xx Blaze</td>
<td>omap4430sdp_config</td>
</tr>
<tr>
<td>3430SDP</td>
<td>omap3430sdp_config</td>
</tr>
<tr>
<td>3630SDP</td>
<td>omap3630sdp_config</td>
</tr>
<tr>
<td>3430LDP</td>
<td>omap3430labrador_config</td>
</tr>
<tr>
<td>Zoom2</td>
<td>omap3430zoom2_config</td>
</tr>
</tbody>
</table>
The resulting u-boot.bin is located in the top-level of your u-boot directory. For Panda board please use omap4430sdp_config

6.2 x-loader
6.2.1 Accessing x-loader source
Clone the x-loader git tree and checkout the source code corresponding to this release:

```
# cd x-loader
# git clone git://git.omapzoom.org/repo/x-loader.git
# git checkout <<commit_ID_for_xloader_in_this_release>>
```

6.2.2 Building x-loader
Make the x-loader source code using the correct config file depending on your platform:

```
# cd x-loader
# make distclean
# make CROSS_COMPILE=arm-none-linux-gnueabi-
# <<config_file_depending_on_your_platform>>
# make CROSS_COMPILE=arm-none-linux-gnueabi- ift
```

6.2.3 x-loader config files

<table>
<thead>
<tr>
<th>Board</th>
<th>config file</th>
</tr>
</thead>
<tbody>
<tr>
<td>44xx Blaze Tablet</td>
<td>omap44XXtablet_config</td>
</tr>
<tr>
<td>44xx Blaze</td>
<td>omap4430sdp_config</td>
</tr>
<tr>
<td>3430SDP</td>
<td>omap3430sdp_config</td>
</tr>
<tr>
<td>3630SDP</td>
<td>omap3630sdp_config</td>
</tr>
</tbody>
</table>
To build x-loader to boot over a serial connection for OMAP4, use "omap4430sdp_config"

The resulting MLO is located in the top-level of your x-loader directory.

Note: If you are using an HS (High Security) OMAP device, an extra step is required. First, build x-load.bin using the steps above. Then, download the MShield signing tool and use the commands below. Contact your TI representative to get access to this tool.

```
# cd mshield-dk-root-folder
# ./generate_MLO <<OMAP type>> x-load.bin
```

For example, for an ES2.3 OMAP4430 device, use the command:

```
# ./generate_MLO OMAP4430 ES2.3 x-load.bin
```

The resulting MLO is located in the top-level of your mshield-dk directory. For up streaming the uboot please refer U-boot Upstreaming Project.

### 6.3 Boot Commands

For booting use the following commands.

Note: The boot commands are already automated through uEnv.txt

```
setenv bootargs 'console=ttyO2,115200n8 androidboot.console=ttyO2
mem=456M@0x80000000 mem=512M@0xA0000000 root=/dev/mmcblk0p2 rw
rootfstype=ext4 ro rootdelay=2 init=/sbin/init vram="16M"
omapfb.vram=0:16M,1:4M omapdss.defDisp=hdmi:1024x600MR-16@60'
```
For network support use

```bash
setenv bootargs 'console=tty02,115200n8 androidboot.console=tty02
mem=456M@0x80000000 mem=512M@0xA0000000 root=/dev/mmcblk0p2 rw
rootfstype=ext4 ro rootdelay=2 init=/sbin/init vram="16M"
omapfb.vram=0:16M,1:4M omapdss.def_disp=hdmi:1024x600MR-16@60 ip=dhcp'
```

And then use

```bash
setenv bootcmd 'mmc rescan 1;fatload mmc 0:1 0x80000000 uImage-omap; bootm
0x80000000'
```

After that use the following command to boot

```bash
Boot
```

U can also save the boot command in uEnv.txt for auto booting.

7 Building the kernel

The latest kernel for OMAP$ can be downloaded from the following link

https://launchpad.net/linaro-landing-team-ti

Use the following commands to build the kernel uImage.

Step 1.

```bash
make ARCH=arm CROSS_COMPILE=Path_to_your/arm-2010q1/bin/arm-none-linux-gnueabi- Tizen_defconfig
```

For more information on Kernel porting please refer to the following link

https://source.tizen.org/sites/default/files/tizen_porting_guide_2.0_alpha.pdf

For reference please refer to the attached configuration file. _tizen_defconfig_

step 2.

```bash
make ARCH=arm CROSS_COMPILE=Path_to_your/arm-2010q1/bin/arm-none-linux-gnueabi- uImage
```
Step 3.
If needed, build and install the kernel modules:

```bash
sudo make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm modules
sudo make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm INSTALL_MOD_PATH=<path to rootfs> modules_install
```

8 Building wireless drivers

- Download WL1271_Linux_SDK 3_00_01_06 package from http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/ecs/WL1271_Linux_SDK/3_00_01_06/index_FDS.html
- Install the SDK

```bash
$ chmod a+x V3.00.01.06-WL6.1.6.0-Linux-x86-Install
$ ./V3.00.01.06-WL6.1.6.0-Linux-x86-Install --mode console
```

The installer will confirm that the SDK will be installed. Enter 'Y' to continue. The default installation path is ~/V3.00.01.06-WL6.1.6.0. Accept the default by pressing Enter.

```
This will install V3.00.01.06-WL6.1.6.0 on your computer. Continue? [n/Y] y
```

Where do you want to install V3.00.01.06-WL6.1.6.0? [/home/testuser/V3.00.01.06-WL6.1.6.0]

```
Installing V3.00.01.06-WL6.1.6.0...
Installing Program Files...
Installation complete.
```

The SDK is installed. Continue with extracting the SDK sources.

```bash
$ cd V3.00.01.06-WL6.1.6.0/software
$ tar xzvf OMAP35x_WL1271_6.1.0.0.144.tar.gz
$ cd MS_TI_OMAP35x_WL1271_6.1.0.0.144/MS_TI_WL1271_Sources
```

Copy the files WLAndroidBuild.sh and wl1271-android.patch into MS_TI_WL1271_Sources directory if the files are extracted into a subdirectory.

- Go to the build directory (if not already done)

```bash
$ cd V3.00.01.06-WL6.1.6.0/software/MS_TI_OMAP35x_WL1271_6.1.0.0.144/MS_TI_WL1271_Sources
```
9 WLAN Calibration

To generate the WLAN calibration NVS file, follow steps listed below. These are based on the steps given here.

- Enable WLAN from Android UI. Ensure that the driver is loaded by checking the messages on the serial console.
- Start the wireless configuration utility wlan_cu on the serial console and enter the commands as follows

Text to be entered on the console is in red colour.

```bash
# wlan_cu -itiwlan0 -b
user_main, start
\> Driver/, Connection/, Management/, Show/, Privacy/, scAn/, roamingG/, qOs/, poWer/, eVents/, Bt coexistance/, Report/, dEbug/, biT/, aboUt, Quit
/ w p l l 2 f 2
\> Driver/, Connection/, Management/, Show/, Privacy/, scAn/, roamingG/, qOs/, poWer/, eVents/, Bt coexistance/, Report/, dEbug/, biT/, aboUt, Quit
.../poWer> set_Power_mode, set_powersave_powerLevel, set_deFault_powerLevel, set_doZe_mode_in_auto, traffic_Thresholds, eNable, Disable, set_dcO_itrIm
/ t r h 0 7
\> Driver/, Connection/, Management/, Show/, Privacy/, scAn/, roamingG/, qOs/, poWer/, eVents/, Bt coexistance/, Report/, dEbug/, biT/, aboUt, Quit
.../biT> Bip/, Radio debug/
.../Radio debug> Get hdk version, cHannel tune, Tx debug/, rx Statistics/
Channel tune of channel 7 was performed OK
/ t b b 3 7 5 1 2 8 0
\> Driver/, Connection/, Management/, Show/, Privacy/, scAn/, roamingG/, qOs/, poWer/, eVents/, Bt coexistance/, Report/, dEbug/, biT/, aboUt, Quit
.../biT> Bip/, Radio debug/
.../Bip> update Buffer calref point, Tx bip, Rx bip/
BufferCalReferencePoint was configured succesfully
/ t b t 1 0 0 0 0 0 0 0
\> Driver/, Connection/, Management/, Show/, Privacy/, scAn/, roamingG/, qOs/, poWer/, eVents/, Bt coexistance/, Report/, dEbug/, biT/, aboUt, Quit
.../biT> Bip/, Radio debug/
.../Bip> update Buffer calref point, Tx bip, Rx bip/
Entering FillMACAddressToNVS
```
Mac[0]=08
Mac[1]=00
Mac[2]=28
Mac[3]=12
Mac[4]=34
Mac[5]=56
exiting FillMACAddressToNVS
/
\> Driver/, Connection/, Management/, Show/, Privacy/, scAn/, roamingG/, qOss/, poWer/, evEnts/, Bt coexsistance/, Report/, deBug/, biT/, aboUt, Quit

 Copy the generated nvs_map.bin to /system/etc/wifi

# mv nvs_map.bin /system/etc/wifi

 Modify wlan_loader service in init.rc as follows:

```
service wlan_loader /system/bin/wlan_loader \
    -e /system/etc/wifi/nvs_map.bin \
    -f /system/etc/wifi/firmware.bin \
    -i /system/etc/wifi/tiwlan.ini
```

disabled
oneshot

With the new settings, when WLAN is enabled, the following log is seen on the console

```
# TIWLAN: driver init
TIWLAN: 1986.373128: wlanDrvIf_Open()
TIWLAN: 1986.463766: pInitParams->RoamingScanning_2_4G_enable 0
SDIO clock Configuration is now set to 24Mhz
TIWLAN: 1986.688985: CHIP VERSION... set 1273 chip top registers
TIWLAN: 1986.700246: Starting to process NVS...
TIWLAN: 1986.704549: NVS found, EEPROM Image addr=0xc8aaca00, EEPROM Len=0x0x1d4
TIWLAN: 1986.712148: Chip ID is 0x4030111.
TIWLAN: 1986.716299: FEM Type 1
TIWLAN: 1986.719534: Starting to download firmware...
TIWLAN: 1986.792226: Starting to download firmware...
TIWLAN: 1986.812399: Starting to download firmware...
TIWLAN: 1986.817403: Starting to download firmware...
TIWLAN: 1986.827993: Starting to download firmware...
```
To make it work with the wpa_supplicant please following the bellow mentioned link


10 Enabling Bluetooth

The procedure to enable the Bluetooth support for Tizen IVI running on PandaBoard (based on TI OMAP 4430 platform) is mentioned below. The procedures mentioned are only tested for PandaBoard Rev 4.

10.1 Kernel Configurations

The following configurations should be enabled inside kernel config file in order to make Bluetooth working.

CONFIG_TI_ST=m
CONFIG_BT_WILINK=m

For RFkill please enable:

CONFIG_RFKILL=y
CONFIG_RFKILL_PM=y
CONFIG_RFKILL_LEDS=y
CONFIG_RFKILL_INPUT=y

Note: Building the Bluetooth drivers as module will save times during booting.
The following kernel modules are needed for Bluetooth to work

1. **st_drv.ko**
2. **btwilink.ko**

For more information regarding the Bluetooth please follow the below mentioned link

http://omappedia.org/wiki/TI_Shared_Transport_Description_and_Support

### 10.2 Board Specific Code Modifications

The below mentioned changes are required for the appropriate GPIO and baud rate settings in the file `board-omap4panda.c` ("Your kernel path"/arch/arm/mach-omap2) please modify the following structures accordingly (if it is different than the below mentioned definitions)

```c
static struct ti_st_plat_data wilink_pdata = {
    .dev_name = "/dev/ttyO1", /* UART2 */
    .nshutdown_gpio = 46, /* BT GPIO may vary from board to board */
    .flow_ctrl = 1,
    .baud_rate = 3686400, /* 115200 for test */
    .suspend = NULL,
    .resume = NULL,
    .chip_enable = NULL,
    .chip_disable = NULL,
    .chip_asleep = NULL,
    .chip_awake = NULL,
};

static struct platform_device wl1271_device = {
    .name = "kim",
    .id = -1,
    .dev.platform_data = &wilink_pdata,
};
```

### 10.3 Compile the kernel and compile the kernel modules

To install the kernel Modules use the following commands:

```
make CROSS_COMPILE=<Your Cross Compiler path>/arm-none-linux-gnueabi- ARCH=arm modules_install INSTALL_MOD_PATH=<Your install path>
```

### 10.4 Downloading the Firmware

The latest firmware files are available on this site:

https://gforge.ti.com/gf/project/wilink_drivers/

Shared Transport driver uses the firmware class sub-system to download firmware files. Hence the firmware ".bts" files should be placed /lib/firmware/for Tizen.
10.5 User Space components

A utility known as UIM is critical for the shared transport (TI_ST) driver to work. (Without UIM, TI_ST will not work.) UIM is a user-space daemon to open the UART.

It can be obtained from:

http://git omapzoom.org/?p=platform/hardware/ti/wpan.git;a=summary

UIM needs to be run at boot since some Linux flavors may require Bluetooth or GPS to be turned on at boot.

To do so (run at boot time), have the UIM entry in one of your .rc files. Alternatively, you can specify a special udev rule based on the platform driver in addition to the device "kim" [Need Modification].

Depending on hardware platform, permission for /dev/ttyxx interface to be mapped to Bluetooth to be added in init.rc

For example please create an udev rule as following

```
# Run every time tibluetooth event is sent by kernel
ACTION="add", SUBSYSTEM="platform", ENV{MODALIAS}"="platform:kim",
RUN="/usr/bin/ti-bt.sh"
```

The script file(ti-bt.sh) mentioned above should look like the following:

```
#!/bin/bash
# TI BT startup script - Description:
# This script is dedicated to start TI BT driver properly.
set +e
/sbin/modprobe st_drv
ps -eaf | grep -v grep | grep "[/ ]uim$" >> /dev/null
if [ "$?" != "0" ]
then
 /usr/bin/uim &
fi
/sbin/modprobe btwilink
exit 0
```

Note:

1. UIM needs access to the UART connected to BT/FM/GPS/NFC and therefore it has to be the owner/group of the UART.
2. The permission for the rfkill entry created by bluetooth kernel sub-system has to be added, which is accessed by bluedroid which belongs to group bluetooth.

10.6 Testing

The Basic hardware functionalities of the Bluetooth on PandaBoard can be tested using the bellow mentioned utilities which are already the part of the Tizen IVI platform.

To check the BT status the following command:

```
$ hciconfig -a
```

To make it up use the following command:

```
$hciconfig hci0 up
```

For More testing use the hcitool utility

hcitool
hcitool - HCI Tool ver 4.101
Usage:
   hcitool [options] <command> [command parameters]
Options:
   --help    Display help
   -i dev    HCI device
Commands:
    dev      Display local devices
    inq      Inquire remote devices
    scan     Scan for remote devices
    name     Get name from remote device
    info     Get information from remote device
    spinq    Start periodic inquiry
    epinq    Exit periodic inquiry
    cmd      Submit arbitrary HCI commands
    con      Display active connections
    cc       Create connection to remote device
    dc       Disconnect from remote device
    sr       Switch master/slave role
    cpt      Change connection packet type
    rssi     Display connection RSSI
    lq       Display link quality
    tpl      Display transmit power level
    afh      Display AFH channel map
    lp       Set/display link policy settings
    lst      Set/display link supervision timeout
    auth     Request authentication
    enc      Set connection encryption
    key      Change connection link key
    clkoff   Read clock offset
    clock    Read local or remote clock
    lescan   Start LE scan
    lewladd  Add device to LE White List
    lewlrn   Remove device from LE White List
    lewlsz   Read size of LE White List
    lewlclr  Clear LE White list
    leccc    Create a LE Connection
    ledc     Disconnect a LE Connection
11 Enabling the sound

For example for Multimedia playback a minimum set of mixer control need to be set:

Note: the below mentioned commands are already incorporated to automatically enable the sound after booting.

Do mkdir /var/lib/alsa in order to store asound state

```
alsactl init
alsactl store/restore
amixer cset name='DL1 Media Playback' 1
amixer cset name='Sidetone Mixer Playback' 1
amixer cset name='Headset Left Playback' 1
amixer cset name='Headset Right Playback' 1
amixer cset name='DL1 PDM Switch' 1
```

After you can set the volume

```
amixer cset name='DL1 Media Playback Volume' 120
amixer cset name='SDT DL Volume' 120
amixer cset name='Headset Playback Volume' 120
```

Then you can make playback on Headset

```
aplay -D plughw:0,0 file.wav
```

12 Enabling the Display

To enable display please modify the /opt/etc/X11/arm-emulfb/display.conf as per the following

```
Section "ServerLayout"
  Identifier "Test Layout"
  Screen 0 "Screen0" 0 0
  # InputDevice "evdev"
EndSection

Section "Screen"
  Identifier "Screen0"
```
Device                  "emulfb"
Monitor                  "Monitor0"
#        SubSection              "Display"
#                Modes           "1024x768"
#        EndSubSection
EndSection

Section "Monitor"
  Identifier              "Monitor0"
#        Option                  "DPMS"  "true"
#       DisplaySize             46 76
EndSection

Section "Device"
  Identifier              "emulfb"
  Driver                  "emulfb"
  Option                  "fbdev"         "/dev/fb0"
# if you need the cursor then do the following, otherwise it is not required
  Option                  "HWCursor"      "off"
  Option                  "SWCursor"      "on"
EndSection

**Note:** Use –nocursor option in /usr/bin/startx in case you do not want the cursor.

Comment out the following portion in /usr/etc/X11/xinitrc if you don’t want the enlightenment to start

# /usr/bin/enlightenment_start

**Note:** Always Connect your HDMI cable after booting the device.

By default the login on the serial debug console is disabled if you want to add it please add the following in your rc.sysinit or rc.network file

/bin/login

Login name: root

Password: Tizen

For more information on panda board bring ups please visit the following link.

[http://omappedia.org/wiki/Main_Page](http://omappedia.org/wiki/Main_Page)