Document Number: AFAQ
Rev kk4.4.2_1.0.0-ga, 07/2014

Android Frequently Asked Questions

1 How do I configure the build information?

For every build, you should define a BUILD ID and BUILD NUMBER. We use our release version as BUILD ID and build user date as BUILD NUMBER.

You can create a buildspec.mk file under your ~/myandroid directory. Android will read this file to get the build information, such as BUILD_ID, BUILD_NUMBER, and default build product, etc.

1. Copy the default buildspec.mk

   ```
   $ cd ~/myandroid
   $  cp build/buildspec.mk.default
   buildspec.mk
   ```

2. Change the buidspec.mk to add BUILD_ID and BUILD_NUMBER. You can add the following lines in buildspec.mk:

   ```
   BUILD_NUMBER := $(USER).$(shell date +%Y%m
   %d.%H%M)
   BUILD_ID :=RXX.XX
   ```

   The BUILD_ID is your software version id. You can change this to your current release number.

3. (Optional) You can also change the TARGET_PRODUCT and TARGET_BUILD_VARIANT in this file to change the default build product.

For example:

```
TARGET_PRODUCT:=sabresd_6dq
TARGET_BUILD_VARIANT:=user
```

4. After these changes, you can directly call `make` to build your Android.

2 How do I download the Android source code behind a firewall?

If you have an HTTPS proxy and your firewall supports socks, perform the following steps:

1. Install Dante, which is a socks client.

```
$ sudo apt-get install dante-client
```

2. Configure Dante by adding below lines into `/etc/dante.conf`.

```
route {
from: 0.0.0.0/0 to: .  via: DNS_OR_IP_OF_YOUR_SOCKS_SERVER port =
PORT_OF_YOUR_SOCKS_SERVER
proxyprotocol: socks_v5
}       resolveprotocol: fake
```

3. Set the environment variable for HTTP proxy and socks.

```
$ export https_proxy=...
$ export SOCKS_USER=...
$ export SOCKS_PASSWD=...
```

4. Download Android code from Google.

```
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ repo init -u https://android.googlesource.com/platform/manifest -b android-4.3_r2.1
$ socksify ~/bin/repo sync
```

3 How do I use ADB over Ethernet?

Since Jelly Bean, security ADB is enabled by default (ro.adb.security is set to 1). In security ADB mode, the ADB over ethernet is not allowed. For more details, see How do I enable and disable security ADB?. To use the ADB over ethernet, there are two steps to follow:

1. Disable the security ADB by changing ADB security setting in init.rc
2. Delete or comment the below line, and then rebuild the boot.img. setprop ro.adb.security 1.
3. Keep the board connecting with usb to the PC, and enable the ‘usb debuging’ from Setting->Developer Options and then follow the steps below to set up ADB over ethernet.

On the Linux PC, assuming you had built Android code or had installed Android SDK), complete the following actions to use ADB over ethernet:

```
 $ ping IP_OF_YOUR_BOARD (run "netcfg" on board to get IP address)
 $ export ADBHOST=IP_OF_YOUR_BOARD
 "adb" is a host tool created during Android build.It's under out/host/linux-x86/bin/. Make
sure you set path properly.
```

```
 $ adb kill-server (Not sure why this step is needed. Just re-start adb daemon on board.)
 $ adb shell
```

Set the ADB port properly on the device:

```
$ setprop service.adb.tcp.port 5555
```

After setting up the ADB listener port, re-enable the USB debug function in the Settings application.

4 How do I set up a computer to support ADB?

In this release, Google vendor ID and product ID are used for all Android gadget functions.

The user can download the latest Android SDK package and use ADB tool to test ADB function.

On the Windows computer, install Google extra win USB driver, contained in the SDK package, when Windows finds your device.

On the Linux computer, add the following rules for udev rule file: /etc/udev/rules.d/51-android.rules

```
SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", MODE="0777"
SUBSYSTEM=="usb|usb_device", ATTR{idVendor}=="18d1", MODE="0666", GROUP="plugdev"
```

5 How do I set up a computer to support ADB in recovery mode?

In this release, Google has added a new way to update the system which supports applying system updates from ADB.

Linux system supports this feature by default if SDK is updated to the version later than Jelly Bean.

For Windows OS, follow the steps below:

1. Install the driver.
2. Apply the patch below to the USB driver from Google
3. Connect the USB cable to the board and install the driver according to the instructions provided.

```
--- android_winusb.inf    2013-06-04 13:39:40.344756457 +0800
+++ android_winusb.inf    2013-06-04 13:43:46.634756423 +0800
@@ -23,6 +23,8 @@


[Google.NTx86]
+;adb sideload support
+%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_D001

;Google Nexus One
%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_0D02
@@ -59,7 +61,8 @@


[Google.NTamd64]
-
+;adb sideload support
+%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_D001
;Google Nexus One
%SingleAdbInterface%        = USB_Install, USB\VID_18D1amp;PID_0D02
%CompositeAdbInterface%     = USB_Install, USB\VID_18D1amp;PID_0D02amp;MI_01
```

6 How do I enable USB tethering?

The USB tethering feature is supported in this release.

The upstream device can be WIFI or Ethernet. USB tethering can be enabled in the Settings UI after your OTG USB cable is connected to computer: Settings -> WIRELESS & NETWORKS -> More.. -> Tethering & portable hotspot -> USB tethering. In the meantime, make sure you have disabled ADB.

On the Linux computer, when USB tethering is enabled, you can easily get an USB network device. The IP and DNS server is automatically configured.

On Windows computer, when you have connected the board with the computer and you can see an unknown device named "Android" in the device manager, you have to manually install the tethering driver by the tetherxp.inf file in android_kk4.4.2_1.0.0-ga_tools.tar.gz. After it is successfully installed, you will see "Android USB RNDIS device" in the device manager. By this time, you can use USB rndis network device to access the network.

7 How do I use MTP?

The Media Transfer Protocol is a set of custom extensions to the Picture Transfer Protocol (PTP).

Whereas PTP was designed for downloading photographs from digital cameras, Media Transfer Protocol supports the transfer of music files on digital audio players and media files on portable media players, as well as personal information on personal digital assistants.

Starting with version 4.0, Android supports MTP as a default protocol transfer files with computer, instead of the USB Mass Storage. In this release, as suggested by Google, we disabled the UMS and enabled MTP.

NOTE

Ensure that you disable the USB Tethering when using MTP. In Windows XP, you cannot make MTP work with ADB enabled. In Windows 7, in theory MTP can work together with ADB, but it is also found that some hosts with Windows 7 fail to support it.

When connecting the board to the computer by USB cable, an USB icon will be shown in the notification bar. Then, you can click on the notification area, and select "Connected as a media device" to launch the USB computer connection option UI. There, MTP and PTP can be chosen as current transfer protocol. You can also launch the option UI by Settings -> Storage -> MENU -> USB computer connection.

MTP on Windows XP

Windows XP supports PTP protocol by default. In order to support MTP protocol, you must install Windows Media Player (Version >= 10). When connecting to a computer, you can see MTP devices in Windows Explorer. Since Windows XP only supports copy/paste files in the explorer, you cannot directly open the files in MTP device.

MTP on Windows 7

Windows 7 supports MTP (PTP) protocol by default. When connecting to a computer, you can see MTP devices in Windows Explorer. You can perform any operations just as you would on your hard disk.

MTP on Ubuntu

Ubuntu supports PTP protocol by default. To support MTP protocol, you must install the following packages: libmtp, mtptools by using the following command:

```
$ sudo apt-get install mtp-tools
```

If your default libmtp version is not 1.1.1 (current latest libmtp on ubuntu is 1.1.0), you must upgrade it manually by:

```
$ sudo apt-get install libusb-dev
$ wget http://downloads.sourceforge.net/project/libmtp/libmtp/1.1.1/libmtp-1.1.1.tar.gz
$ tar -xvf libmtp-1.1.1.tar.gz
$ cd libmtp-1.1.1
$ ./configure --prefix=/usr
$ make
$ sudo make install
```

After you have done the steps outlined above, you can transfer the files between the computer and the device by using the following commands:

- mtp-detect finds current connected MTP device.
- mtp-files lists all the files on MTP device.
- mtp-getfile gets the files on MTP device by file ID listed by mtp-files.
- mtp-sendfile puts files onto MTP device.

There's an alternative GUI application, called gMtp, which makes it easier to access MTP device instead of using the commands above. You can install it by using the following command:

```
$ sudo apt-get install gmtp
```

After installation, you can launch gmtp and access MTP device in the file explorer.

8 How do I enter the Android recovery mode manually?

This process only works on i.MX 6Quad SD v1 board.

Press "**VOLUME** -" and "**Power**" to enter Recovery mode. This check is in u-boot.git board support file, where you can change your preferred combo keys.

Also, you can input this command in the kernel:

```
# reboot recovery             # the board reset to recovery mode.
```

to enter recovery mode.

8.1 How do I enter the text menu in recovery mode?

To enter the text menu in recovery mode, follow the steps below:

1. When the system completes bootup, an Android Robot Logo displays.
2. Press the POWER KEY (keep pressed), and use the VOLUME UP KEY as follows:

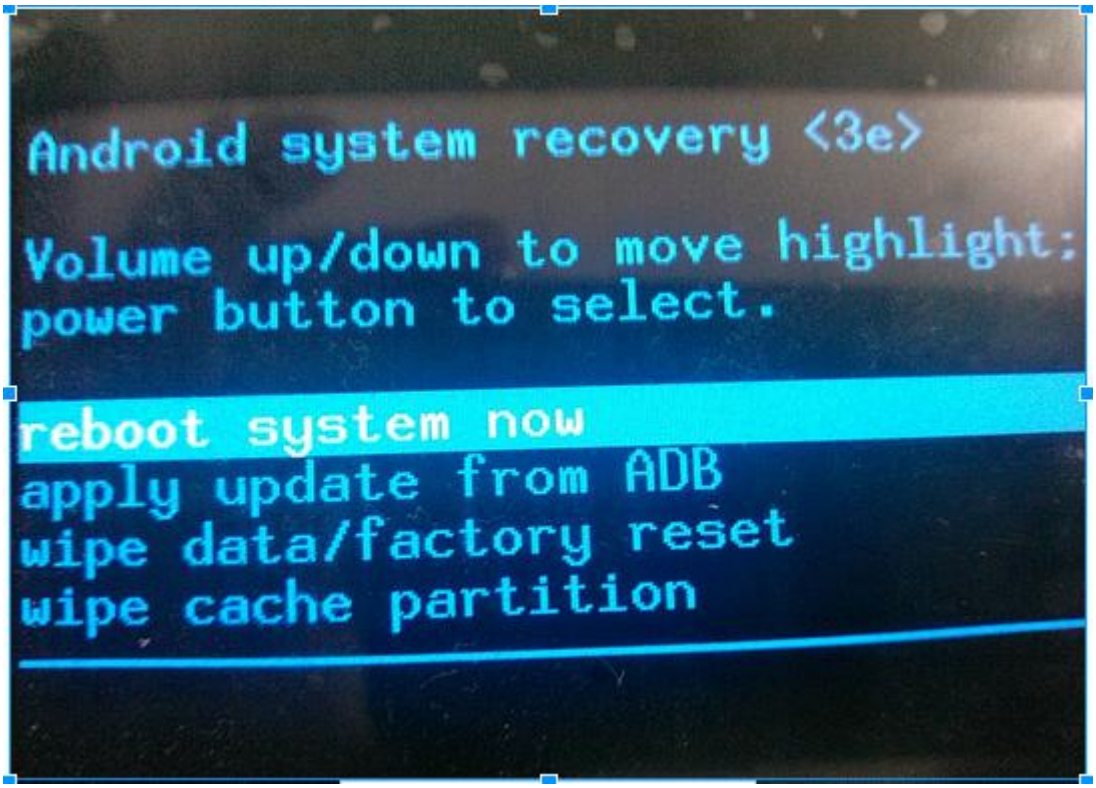

Figure 1. How do I enter the Text menu in recovery mode

3. Move the menu item by VOLUME UP and VOLUME DOWN button.
4. Select the menu item by Power Key.
5. Select the required option using the direction keys on the keypad or keyboard.
6. Reboot the system
7. Apply updates from ADB. Note that you may update the software from update.zip by using the adb sideload command.
8. Wipe data/factory reset. /data and /cache partitions are formatted.
9. Wipe cache partition. /cache partition is formatted.
10. Reboot the system.

8.2 How do I use Android fastboot?

Fastboot is a feature which can be used to download images from Windows/Linux computer to the target storage device.

This feature is released by Google in the Android SDK package, which can be downloaded from Android official site. Freescale Android release implements part of the fastboot commands in U-Boot such as: flash, reboot, getvar.

Before using fastboot, Android SDK must be installed on the host, and the target board must boot up to bootloader. Also, before using fastboot, U-Boot must be downloaded to the MMC/SD device with all the partitions created and formatted. Setup the correct board dip switches to boot up the board with U-Boot.

NOTE

The size of images downloaded by fastboot must be < 320 MB.

Target side:

1. Power on the board with USB OTG connected.
2. Press any key to enter the U-Boot shell.
3. Select the correct device to do fastboot image download by command:
   - SD/MMC card

```
U-Boot > setenv fastboot_dev mmc3
```

- Run the fastboot command:

```
U-Boot > fastboot
              fastboot is in init......USB Mini b cable Connected!
              fastboot initialized
              USB_SUSPEND
              USB_RESET
              USB_RESET
```

Or launch the quick fastboot by "fastboot q" command

```
U-Boot > fastboot q
fastboot is in init......flash target is MMC:1
USB Mini b cable Connected!
```

Or you can input this command in the kernel:

```
# reboot fastboot              # the board reset to fastboot mode.
```

All commands can enter fastboot mode.

NOTE

- On host computer, it will prompt you that a new device is found and that you need to install the driver. You need to install it.
- The quick fastboot is a new implementation for fastboot utility. It increases the image download speed from computer to board up to about 28 MB/s, compared to the previous 1 MB/s. It only supports download and flash command now, which indicates that it supports image download.

Host side:

1. Enter the Android SDK tools directory and find the fastboot utility (fastboot.exe on Windows, fastboot on Linux).
2. Copy all downloaded images to the "images" folder.
3. Run the following commands to flash the SD or eMMC:

```
$ fastboot flash bootloader images\u-boot-no-padding.bin
$ fastboot flash boot images\boot.img
$ fastboot flash system images\system.img
$ fastboot flash recovery images\recovery.img
$ fastboot reboot
```

4. Run the following commands to flash NAND

```
$ fastboot flash boot images\boot.img
$ fastboot flash android_root images\android_root.img
$ fastboot flash recovery images\recovery.img
$ fastboot reboot
```

NOTE

Fastboot does not support flashing the bootloader to NAND storage.

8.3 How do I update the system by using ADB?

Before upgrading the system with the ADB tool windows users first need to install the ADB driver. To do so, please refer to How do I setup PC to support ADB In Recovery mode section.

After the installation and set up of the driver is complete, follow the steps below:

1. Ensure that the system has entered recovery mode.
2. Refer to the How Do I enter Android Recovery Mode and toggle the text Menu.
3. Move the cursor to “Apply update from ADB.” The UI will display as shown below:

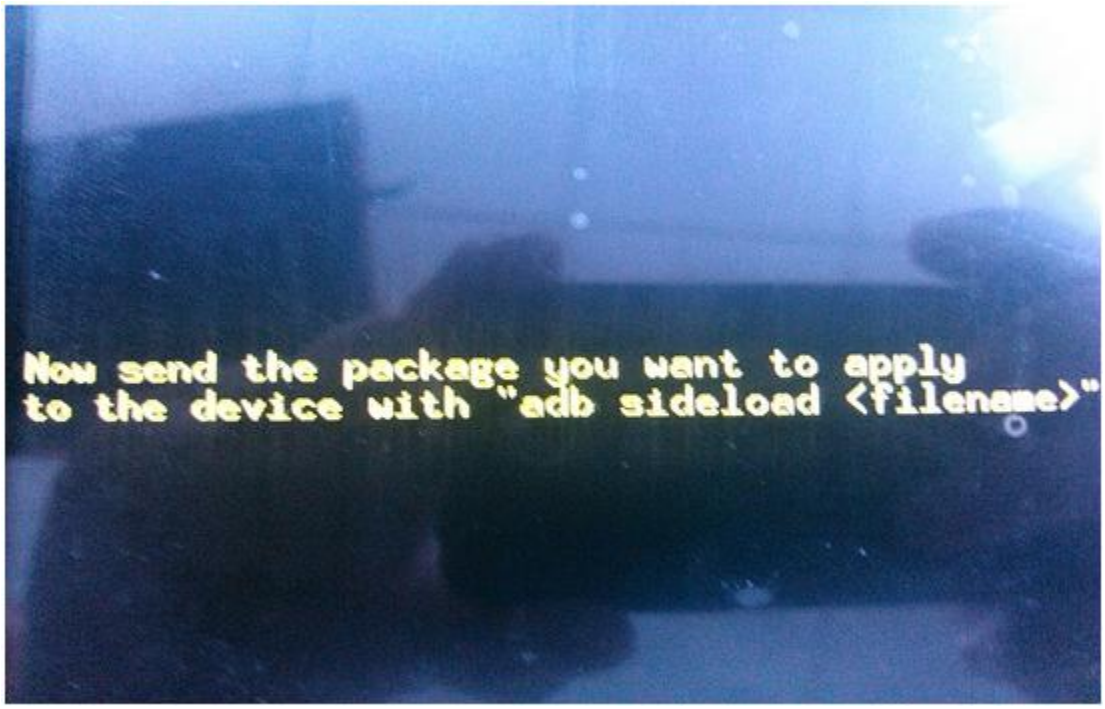

Figure 2. How to upgrade system by ADB

4. On your PC, execute below cmds
5. `adb sideload $YOUR_UPDATE_PACKAGE.zip`
6. After the sending is completed,the system starts updating the firmware with the update file.

9 What is the key mapping of the USB keyboard?

The default DELL USB keyboard key mapping is defined as shown below.

Key	Act as
ESC	BACK
F1	MENU
F2	SOFT_RIGHT
F3	CALL
F4	ENDCALL
F5	ENDCALL
F8	HOME
F9	DPAD_CENTER
UP	DPAD_UP
DOWN	DPAD_DOWN

Table continues on the next page...

Key	Act as
BACK	DEL
ENTER	ENTER

10 What is the key mapping of onboard keypad in i.MX6SoloLite EVK board?

The key mapping of the default onboard keypad is defined as below:

Key	Act as
SW6	VOLUME+
SW7	VOLUME-
SW11	POWER
SW12	BACK
SW10	DPAD_UP
SW13	DPAD_DOWN
SW9	DPAD_LEFT
SW8	DPAD_RIGHT

11 How do I generate uramdisk.img?

To generate a RAMDISK image recognized by U-Boot, perform the following operations:

NOTE

Uramdisk is not used anymore.

Assume you have already built U-Boot and mkimage is generated under myandroid/bootable/bootloader/uboot-imx/tools/.

```
$ cd myandroid/out/target/product/sabresd_6q
$ ~/myandroid/bootable/bootloader/uboot-imx/tools/mkimage
-A arm -O linux -T ramdisk -C none -a 0x10308000 -n "Android Root Filesystem" -d ./
ramdisk.img
./uramdisk.img
```

12 How do I generate boot.img?

Use the following commands to generate boot.img

```
$ mkbootimg --kernel <kernel, zImage> --ramdisk < ramdisk> --base < baseaddr> --cmdline
<kernel
command line> --board < board name > -o <output>
$ cd myandroid
$ out/host/linux-x86/bin/mkbootimg --kernel kernel_imx/arch/arm/boot/zImage --ramdisk
ramdisk.img --base 0x10800000 --cmdline "console=ttymxc0,115200 init=/init rw video=mxcfb0
vmalloc=400M" --board mx6q_sabrelite -o boot.img
```

NOTE

If you want to extract and edit the zImage and ramdisk in the boot.img, see www.android-dls.com/wiki/index.php?title=HOWTO:_Unpack%2C_Edit%2C_and_Re-Pack_Boot_Images.

13 How do I change the boot command line in boot.img?

After using boot.img, we store the default kernel boot command line inside this image.

It will package together during Android build.

You can change this by changing BOARD_KERNEL_CMDLINE which is defined in android/{product}/BoardConfig.mk file.

NOTE

Replace {product} with your product, such as sabresd_6q.

14 How do I customize the boot animation?

The user can create his/her own boot animation for his/her device.

Android provides an easy way to replace its default boot animation by putting bootanimation.zip file into /system/media/.

- To create your own bootanimation .zip file, see www.addictivetips.com/mobile/how-to-change-customize-create-android-boot-animation-guide.
- How to install the boot animation
  - On the host, use ADB to download the bootanimation.zip file into the device, for example:

```
$ adb push ~/Downloads/bootanimation.zip /mnt/sdcard
```

  - On the device, remount the /system to writable, and copy the file:

```
$ mount -t ext4 -o rw,remount /dev/block/mmcblk0p5 /system
$ busybox cp /mnt/sdcard/bootanimation.zip /system/media/
$ mount -t ext4 -o ro,remount /dev/block/mmcblk0p5 /system
```

15 Why cannot certain APKs run on a device without a modem?

Some games require the TelephonyManager to return a device ID by getDeviceId() method of TelephonyManager, which is actually to return the mobile IMEI code with modem connected, but **null** without a modem.

They do not check the return value of getDeviceId(). Therefore, you can probably use null as device id without doing NULL as shown below:

```
TelephonyManager localTelephonyManager =
(TelephonyManager)oAppMain.getSystemService("phone");
      String str;
      if (localTelephonyManager != null)
      {
        nativeAddProperty("IMEI", localTelephonyManager.getDeviceId());
        Object[] arrayOfObject = new Object[1];
        arrayOfObject[0] = Integer.valueOf(Integer.parseInt(Build.VERSION.SDK));
        nativeAddProperty("DeviceID", String.format("%d", arrayOfObject));
        str = oAppMain.getClass().getPackage().getName();
```

```
        nativeAddProperty("Identifier", str);
      }
```

On the platform, when there is no modem connected, the TelephonyManager.getDeviceId() will return null. So, the JNI calling from here nativeAddProperty crashes in the dalvik JNI module, which will try to get strings from a null pointer.

For the tablet customer who does not have a modem connected, a workaround may need to be applied into framework/base.git:

```
diff --git a/telephony/java/android/telephony/TelephonyManager.java
b/telephony/java/android/telephony/TelephonyMa
index db78e2e..82cf059 100755
--- a/telephony/java/android/telephony/TelephonyManager.java
+++ b/telephony/java/android/telephony/TelephonyManager.java
@@ -192,6 +192,9 @@ public class TelephonyManager {
      *   {@link android.Manifest.permission#READ_PHONE_STATE READ_PHONE_STATE}
      */
     public String getDeviceId() {
+       String s = "2222222222";
+       return s;
+       /*
         try {
             return getSubscriberInfo().getDeviceId();
         } catch (RemoteException ex) {
@@ -199,6 +202,7 @@ public class TelephonyManager {
         } catch (NullPointerException ex) {
             return null;
         }
+       */
     }
```

To verify your IMEI hard code, start the phone application and dial *#06#. It will show the IMEI code.

16 How do I enable or disable the bus frequency feature?

The Bus Frequency driver is used to slow down DDR, AHB, and AXI bus frequency in the SoC when the IPs which need high bus frequency are not working.

This saves the power consumption in Android earlysuspend mode significantly (playing audio with screen off). In this release, the bus frequency driver is **enabled** by default. If you want to enable or disable it, perform the following command in the console:

```
Disable:
$ echo 0 > /sys/devices/platform/imx_busfreq.0/enable
Enable:
$ echo 1 > /sys/devices/platform/imx_busfreq.0/enable
```

Note that if you are using Ethernet, the up operation will enable the FEC clock and force bus frequency to be high. That means you cannot go into low bus mode anymore, regardless whether the Ethernet cable is plugged or unplugged. Therefore, if you want the system to go into the low bus mode, you must do 'netcfg eth0 down' to shut down the FEC manually. If you want to use FEC again, do 'netcfg eth0 up' manually. When FEC is shut down with clock gated, the PHY cannot detect your cable in/out events.

17 How to set networking proxy for Wi-Fi?

To configure the proxy settings for a Wi-Fi network, do as follows:

1. Tap and hold a network from the list of added Wi-Fi networks.
2. Select "Modify Network".

3. Choose "Show advanced options".
4. If no proxy settings are present in the network, you have to - Tap "None", Select "Manual" from the menu that opens.
5. Enter the proxy settings provided by the network administrator.
6. Finally tap on the button denoted as "Save"

18 How do I enable 512MB DDR memory in U-Boot?

The default DDR memory configuration is "1GB" on i.MX 6Dual/6Quad and i.MX 6DualLite board.

The Freescale Android platform recommends to adopt 1GB memory by default. If you want to enable the 512MB memory, you can refer to the following modifications:

```
~/myandroid/bootable/bootloader/uboot-imx/include/configs/mx6dl_sabresd.h
~/myandroid/bootable/bootloader/uboot-imx/include/configs/mx6q_sabresd.h

-#define PHYS_SDRAM_1_SIZE      (1u * 1024 * 1024 * 1024)
+#define PHYS_SDRAM_1_SIZE      (1u * 512 * 1024 * 1024)

~/myandroid/bootable/bootloader/uboot-imx/include/configs/mx6dl_sabresd_android.h
~/myandroid/bootable/bootloader/uboot-imx/include/configs/mx6q_sabresd_android.h

-               "splashimage=0x30000000\0"                              \
+               "splashimage=0x1D000000\0"                              \

~/myandroid/device/fsl/sabresd_6dq/BoardConfig.mk (change the default boot commands)
-BOARD_KERNEL_CMDLINE := console=ttymxc0,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off
video=mxcfb2:off fbmem=10M fb0base=0x27b00000 vmalloc=400M androidboot.console=ttymxc0
androidboot.hardware=freescale
+BOARD_KERNEL_CMDLINE := console=ttymxc0,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off
video=mxcfb2:off fbmem=10M fb0base=0x17b00000 gpumem=96M vmalloc=400M
androidboot.console=ttymxc0 androidboot.hardware=freescale
```

NOTE

To run Android platform with 512MB DDR memory, you have to configure GPU memory as the smaller value which can allow more free memory in the system. Meanwhile, the fb0base should also be adjusted. When you build the software, ensure to include “~/myandroid/device/fsl/sabresd_6dq/BoardConfig.mk” changes, which will build-in the boot commands in your boot.img. If you use the release image, set the correct boot command manually as follows:

```
setenv bootargs console=ttymxc0,115200 init=/init
video=mxcfb0:dev=ldb,bpp=32 video=mxcfb1:off video=mxcfb2:off
fbmem=10M fb0base=0x17b00000 gpumem=96M vmalloc=400M
androidboot.console=ttymxc0 androidboot.hardware=freescale
```

19 How do I run the image on i.MX 6Solo board?

The user can adopt the correct bootloader u-boot.bin so that the image can run on i.MX 6Solo board.

One emulation configuration for i.MX 6Solo has been provided in the default U-Boot delivery (mx6solo_sabresd_android_config) to ensure that DDR is 32 bit. The user can refer to this configuration and follow the instructions in User Guide 'Build U-Boot Image’ to build the i.MX 6Solo u-boot.bin.

```
For i.MX6solo SabreSD:
make mx6solo_sabresd_android_config
```

NOTE

To emulate single core, the user can add "nosmp" to the boot command line to disable SMP:

```
setenv bootargs console=ttymxc0,115200 init=/init nosmp
video=mxcfb0:dev=ldb,bpp=32 video=mxcfb1:off video=mxcfb2:off
gpumem=96M
fbmem=10M fb0base=0x17b00000 vmalloc=400M
androidboot.console=ttymxc0 androidboot.hardware=freescale
```

The user can also disable the configuration "CONFIG_SMP" in the kernel.

20 How do I enable waking up system by using the USB HID device?

The Android framework supports wake up of the whole system by the external input device such as USB, BT, or HID.

If the user wants to enable this feature, the user needs to enable the USB remote wake up feature in kernel by default. Applying the following patches to the kernel enables USB remote wake up for OTG port and all the connected devices:

```
diff --git a/arch/arm/plat-mxc/usb_common.c b/arch/arm/plat-mxc/usb_common.c
index 97d963a..bc10b88 100755
--- a/arch/arm/plat-mxc/usb_common.c
+++ b/arch/arm/plat-mxc/usb_common.c
@@ -484,7 +484,7 @@ static int usb_register_remote_wakeup(struct platform_device *pdev)
           return -ENODEV;
    }
    irq = res->start;
-   device_set_wakeup_capable(&pdev->dev, true);
+   device_init_wakeup(&pdev->dev, true);
    enable_irq_wake(irq);
    return 0;
diff --git a/drivers/usb/core/hub.c b/drivers/usb/core/hub.c
index 55a63e4..b7da7b0 100644
--- a/drivers/usb/core/hub.c
+++ b/drivers/usb/core/hub.c
@@ -1546,7 +1546,7 @@ void usb_set_device_state(struct usb_device *udev,
           recursively_mark_NOTATTACHED(udev);
    spin_unlock_irqrestore(&device_state_lock, flags);
    if (wakeup >= 0)
-           device_set_wakeup_capable(&udev->dev, wakeup);
+           device_init_wakeup(&udev->dev, wakeup);
}
EXPORT_SYMBOL_GPL(usb_set_device_state);
```

21 How do I configure the logical display density?

The Android UI framework defines a set of standard logical densities to help the application developers target application resources. Device implementations MUST report one of the following logical Android framework densities:

- 120 dpi, known as 'ldpi'
- 160 dpi, known as 'mdpi'
- 213 dpi, known as 'tvdpi'
- 240 dpi, known as 'hdpi'
- 320 dpi, known as 'xhdpi'
- 480 dpi, known as 'xxhdpi'

Device implementations SHOULD define the standard Android framework density that is numerically closest to the physical density of the screen, unless that logical density pushes the reported screen size below the minimum supported. To configure the logical display density for framework, you must define the following line in the init.freescale.rc:

```
setprop ro.sf.lcd_density <density>
```

22 How do I enable and disable security ADB?

The versions later than Android JB 4.2.2 introduce public key authentication policy when connecting to PC host via ADB.

This feature will keep ADB offline until the device and PC pass authentication. With security ADB enabled, Android SDK Platform-tools in PC host side higher than revision 16.0.2 is needed to make a valid ADB connection. Otherwise, PC can't connect the device over ADB(the status will always be offline). With latest Platform-tools, the device will pop-up a dialog to remind you to allow ADB connect or not when connecting the device with PC through ADB. You need select "OK" to allow PC connect your devices. If select "Cancel", the ADB connect will be rejected. For running CTS, you need to select the "Always allow from this computer". Otherwise, the device will still need your confirmation in each reboot. And to disable it please delete this line or comment it out as below:

```
#setprop ro.adb.secure 1
```

23 How do I enable BT on i.MX 6 SABRE-SD Rev. C?

The versions later than Android JB 4.2.2 introduce bluedroid infrastructure instead of bluez.

To enable BT on i.MX 6 SABRE-SD Rev C, see the following document on freescale community website: www.community.freescale.com/docs/DOC-94235

24 How do I change BT MAC address on i.MX 6 SABRE-SD Rev C board?

If you enable BT on i.MX 6 SABRE-SD Rev C board and find that certain issues are caused by the fact that the same MAC address is shared by two BT peers, you could do the following to change BT MAC address.

1. cat /system/etc/firmware/ar3k/1020200/ar3kbdaddr.pst

   Check the original MAC address. The result looks like this: 1260417f0300(12:60:41:7f:03:00).

2. Mount -w -o remount /system

   Change system file system as writable.

3. echo "*******" > /system/etc/firmware/ar3k/1020200/ar3kbdaddr.pst
4. It is changed.

25 How do I configure rear and front camera?

Property "back_camera_name" and "front_camera_name" are used to configure the camera being used as rear camera or front camera. The name should be either v4l2_dbg_chip_ident.match.name returned from v4l2's IOCTL VIDIOC_DBG_G_CHIP_IDENT or v4l2_capability.driver returned from v4l2's IOCTL VIDIOC_QUERYCAP. Camera

HAL will go through all the V4L2 device present in system. Camera HAL will choose the first matched name in property setting as the corresponding camera. Comma is used as a delimiter of different camera names among multiple camera selections.

Below is an example been set in myandroid/device/fsl/sabresd_6dq/init.rc.

setprop back_camera_name ov5640_mipi

setprop front_camera_name uvc,ov5642_camera,ov5640_camera.

media_profiles.xml in /system/etc is used to configure the parameters been used in recording video and taking picture. Freescale provide several media profile examples which help customers to make the parameters align with their camera module's capability and their device definition.

Table 1. Camera settings

Profile file name	Rear camera	Front camera
media_profiles_1080p.xml	maximum to 1080P, 30FPS and 8Mb/s for recording video	maximum to 720P, 30FPS, and 3Mb/s for recording video
media_profiles_720p.xml	maximum to 720P, 30FPS, and 3Mb/s for recording video	maximum to 720P, 30FPS, and 3Mb/s for recording video
media_profiles_480p.xml	maximum to 480P, 30FPS, and 2Mb/s for recording video	maximum to 480P, 30FPS, and 2Mb/s for recording video
media_profiles_qvga.xml	maximum to QVGA, 15FPS, and 15Mb/s for recording video	maximum to QVGA, 15FPS, and 15Mb/s for recording video

NOTE

Since not all uvc cameras can have 1080P, 30 FPS resolution setting, it is recommended that media_profiles_480p.xml be used for any board's configuration which defines the uvc as rear camera or front camera.

26 How do I configure camera sensor parameters?

Since Android Jelly Bean 4.3, it needs camera sensor focal length and sensitive element size to calculate view angle when using panorama. The focal length and sensitive element size should be customized based on the camera sensor being used. The default release has the parameters for ov5642/ov5640 as front/back camera.

The Ov5640Csi.*/Ov5640Mipi.*/Ov5642Csi.* in hardware/imx/mx6/libcamera2 are provided to configure sensor. They implement class Ov5640Mipi/Ov5642Csi/Ov5640Csi. For a new camera sensor, a new camera sensor class should be created with the corresponding focal length and sensitive element size as the variables mFocalLength, mPhysicalWidth, and mPhysicalHeight in the class.

Table 2. Camera sensor parameters

Parameters	Description
mFocalLength	mFocalLength
mPhysicalWidth	sensitive element width
mPhysicalHeight	sensitive element height

27 How do I configure UBI image for miscellaneous NAND chips?

The default UBI image built for SabreAuto board is configured for the NAND chip MT29F8G08ABABAWP with below attribution:

- Minimum input/output unit size of the flash is 4096
- Logical eraseblock (LEB) size of the UBI volume is 516096
- Physical eraseblock size of the flash chip is 512Kib

Above information can be fetched through below command or related NAND chip's specification:

```
$ mtdinfo /dev/mtd3 -u
mtd3
Name:                            user
Type:                            nand
Eraseblock size:                 262144 bytes, 256.0 KiB
Amount of eraseblocks:           3456 (905969664 bytes, 864.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                   4096 bytes
OOB size:                        224 bytes
Character device major/minor:    90:6
Bad blocks are allowed:          true
Device is writable:              true
Default UBI VID header offset:   4096
Default UBI data offset:         8192
Default UBI LEB size:            253952 bytes, 248.0 KiB
Maximum UBI volumes count:       128
```

Below is an example to make the UBI image for the NAND MT29F8G08ABACAWP, which has the following attribution:

- minimum input/output unit size of the flash is 4096
- logical eraseblock size of the UBI volume is 253952
- physical eraseblock size of the flash chip is 256Kib

```
$~/myandroid/device/fsl$ git diff
diff --git a/sabreauto_6q/BoardConfig.mk b/sabreauto_6q/BoardConfig.mk
index b118ec5..f3db72c 100644
--- a/sabreauto_6q/BoardConfig.mk
+++ b/sabreauto_6q/BoardConfig.mk
@@ -85,8 +85,8 @@ BOARD_KERNEL_CMDLINE := console=ttymxc3,115200 init=/init
video=mxcfb0:dev=ldb,b
 ifeq ($(TARGET_USERIMAGES_USE_UBIFS),true)
 #UBI boot command line.
 UBI_ROOT_INI := device/fsl/sabreauto_6q/ubi/ubinize.ini
-TARGET_MKUBIFS_ARGS := -m 4096 -e 516096 -c 4096 -x none
-TARGET_UBIRAW_ARGS := -m 4096 -p 512KiB $(UBI_ROOT_INI)
+TARGET_MKUBIFS_ARGS := -m 4096 -e 253952 -c 4096 -x none
+TARGET_UBIRAW_ARGS := -m 4096 -p 256KiB $(UBI_ROOT_INI)
```

NOTE

This NAND partition table must align with MFGTool's config.
BOARD_KERNEL_CMDLINE += mtdparts=gpmi-nand:16m(bootloader),
16m(bootimg),128m(recovery),-(root) ubi.mtd=3.

28 How do I enable developer settings on Android Jelly Bean and later version?

Google has hidden the developer settings since the latest version of Jelly Bean. The following steps explain how to retrieve developer settings:

- Go to the settings menu and scroll down to "About phone." Tap it.
- Scroll down to the bottom again until you see "Build number."
- Tap it seven (7) times. After the third tap, you'll see a playful dialog that says you're four taps away from being a developer.
- Keep on tapping, until you've got the developer settings back.

29 How do I customize the U-Boot splash screen?

Splash screen is supported to make the screen display something quickly after power-on in U-Boot. The general data flow of this function is "bmp file -> ipuv3 framebuffer -> ipuv3 display module -> LDB(LVDS Display Bridge) -> LVDS panel". Currently, we only support splash screen on one LVDS panel, i.e., LDB works in a separate channel mode (one of the two channels is enabled) and one IPU DI is enabled. User may choose to display the splash image on either LVDS interface 0 or LVDS interface 1. As LDB works in a separate channel mode, it cannot support high resolution display such as 1080P@60 Hz LVDS panel. To enable this kind of display, a customer may change LDB mode to a split mode. Dual LVDS panel splash screen can be supported by changing the LDB mode to a dual mode or a separate mode. In this case, the two LVDS display panels should have the same video mode. Normally, LDB separate channel mode is enough for an ordinary mobile pad or phone whose display device is a single LVDS panel and its video mode is up to 1024x768@60 Hz or 1280x800@60 Hz.

Here are some pointers the customer should keep in mind when customizing the single LVDS display with LDB separate mode:

- Set the PIN IOMUX setting for backlight and any other necessary pin setting or regulator setting according to specific board design.
- Enable PWM backlight. Use the correct PWM module according to a specific board design.
- To specify the display video mode, change the definition of struct fb_videomode variable ‘lvds_xga’ in ‘board/freescale/mx6q_sabresd/mx6q_sabresd.c’.
- Configure the pixel format on LVDS data bus by changing the interface_pix_fmt parameter when calling ipuv3_fb_init().

Change the following clock trees to get the IPU hsp clock and the pixel clock of LVDS panel:

- IPU hsp clock for i.MX 6Quad Sabre-SD:
  - osc_clk(24 M) -> pll2_528_bus_main_clk(528 M) -> periph_clk(528 M) -> mmdc_ch0_axi_clk(528 M) -> ipu1_clk(264 M)
- IPU hsp clock for i.MX 6DualLite Sabre-SD:
  - osc_clk(24 M) -> pll3_usb_otg_main_clk(480 M) -> pll3_pfd_540(540 M) -> ipu1_clk(270 M)
- IPU pixel clock for i.MX 6Quad Sabre-SD and i.MX 6DualLite Sabre-SD:
  - osc_clk(24 M) -> pll2_528_bus_main_clk(528 M) -> pll2_pfd_352M(452.57 M) -> ldb_dix_clk(64.65 M) -> ipu1_di_clk_x(64.65 M) -> ipu1_pixel_clk_x(64.65 M)

To support smooth UI transition from U-Boot to kernel, the clock trees used in U-Boot should be aligned with the kernel clock trees. The fb0Base is used to tell kernel the framebuffer0 base address, which is shared with U-Boot for splash screen function. The Linux Kernel will reuse fb0Base for framebuffer0. On i.MX 6Dual6Quad and i.MX 6DualLite platform, the GPU2D/3D cores are used to accelerate graphic performance. The fb0Base address is accessed by GPU. The GPU 3D/2D cores can directly access the physical memory if the address is lower than 2G. The GPU3D/2D core achieves a better performance for memory with the physical address which is lower than 2 G than memory with the physical address above 2

G. Please note that the fb0Base address should not exceed the GPU allowed address boundary. This means that you should not set the fb0Base larger than 0x8000000. Otherwise, the performance will drop due to the address limitation of GPU2D/3D hardware.

30 How do I reduce the RTSP streaming latency?

The OMXPlayer has a cache buffer 4 seconds of data in size. This buffer introduces some latency for audio/video streaming. If you want to reduce the latency, you can modify the code below to control the cache size.

```
In file: myandroid/external/fsl_imx_omx/OpenMAXIL/src/component/streaming_parser/
StreamingParser.cpp
    #define PACKET_CACHE_SIZE (4*OMX_TICKS_PER_SECOND)
```

This feature is available as part of the Freescale Extended Multimedia Feature Package. Please send inquiry to L2manager-android@freescale.com for more information and details about the package.

31 How do I set GPU minimal clock to balance performance and power consumption?

Normally GPU works at full speed. When thermal driver reports that the chip is too hot, the GPU driver adjusts the internal clock to reduce the power consumption and quickly cool down the chip. In theory the GPU clock should be set to 1/64 to ensure that chip can cool down faster. However, you may see a black screen or experience a flickering issue when the GPU works with such a slow clock especially in large resolutions (for example 1080P).

The steps below show how to customize the threshold of the GPU minimal clock based on the chip (i.MX 6Quad or i.MX 6DualLite) and the resolution of their product.

Customer can set the minimal GPU clock by adding the line below in init.rc file. The value can be set to any value from 1 to 64. Current default value is 1. The recommended value is 3 on i.MX 6Quad and 8 on i.MX 6DualLite for 1080 p display. A customer should tune and set a suitable value based on their test.

Write

```
/sys/module/galcore/parameters/gpu3DMinClock 1
```

32 How to enable Wi-Fi Display feature by using a Wi-Fi module of other vendor?

Basically, there are two roles in Miracast connection. One is Miracast source device; the other is Miracast sink device. The TDLS&Wifi direct can be used to establish the Miracast connection. The Wi-Fi direct is a mandatory feature for Miracast. TDLS is optional. By default, Realtek Wi-Fi module (rtl8723as) is supported in our formal release that can support Wi-Fi display. If you want to use other Wi-Fi modules from Realtek or from other vendors like Atheros, BCM, and CSR, you should ensure that it supports the following features:

- Two interfaces concurrent mode.
- Supporting AOSP Wi-Fi display configuration, not using their own private way to support it which will provide standard interfaces to change the role of Miracast connection.
- Supporting 802.11n. Data throughput cannot be less than 30 Mbps.

33 How do I check frame drop statistic during video playback?

Input the commands below from console to enable the frame drop statistics for video playback.

```
> echo 3 > /data/omx_log_level
> chmod 777 /data/omx_log_level
```

Then check the frame drop statistic with logcat which displays as follows:

Total frames: 6098, Total Dropped frames: 0, Render device dropped frames: 0

Total frames: The total frames of the video file. Since drop B frame is enabled by default for performance tuning, and is not included in the total frame calculation, so the total frame in the frame drop statistic may not equal to the file real total frame count.

Total Dropped frames: The dropped frame count as AV synchronization.

Render device dropped frames: The dropped frame count in surface texture.

34 How do I build an OTA package?

The Android build system supports auto generation of update.zip. It can generate the updater_script and all system.img files.

You can use the following command to generate an OTA package.

```
$ make dist
```

For example, you use this command to build sabresd_6dq product after the build finish.

```
$ make PRODUCT=sabresd_6dq-eng dist -j4
```

You can find a few packages in the following path for the OTA update.zip to do the OTA and make a different package.

- out/dist/sabresd_6dq-ota-eng.xxx.zip
- out/target/proudct/sabresd_6dq/sabresd_6dq-ota-eng.xxx.zip

The package we used before R13.1 was update.zip. In this package, however, everything is automatically generated.

35 How do I sign the OTA package with my digital keys?

Android requires that each application be signed with the developer's digital keys to enforce signature permissions and application request to use shared user ID or target process. For more information on the general Android security principles and signing requirements, see Section "Android Security and Permissions" in the *Android Developer Guide*. The core Android platform uses four keys to maintain security of core platform components:

- **platform**: a key for packages that are part of the core platform.
- **shared**: a key for content shared in the home/contacts process.
- **media**: a key for packages that are part of the media/download system.
- **releasekey**: default key to sign with if nothing specified.

How do I sign the OTA package with my digital keys?

These keys are used to sign applications separately for release images and are not used by the Android build system. The build system signs packages with the testkeys provided in build/target/product/security/. Because the testkeys are part of the standard Android open source distribution, they should never be used for production devices. Instead, device manufacturers should generate their own private keys for shipping release builds.

Generating keys

A device manufacturer's keys for each product should be stored under vendor/<vendor_name>/security/<product_name>, where <vendor_name> and <product_name> represent the manufacturer and product names. To simplify key creation, copy the script below to this directory in a file called mkkey.sh. To customize your keys, change the line that starts with AUTH to reflect the correct information for your company:

```
#!/bin/sh
AUTH='/C=US/ST=California/L=Mountain View/O=Android/OU=Android/CN=Android/
emailAddress=android@android.com'
if [ "$1" == "" ]; then
        echo "Create a test certificate key."
        echo "Usage: $0 NAME"
        echo "Will generate NAME.pk8 and NAME.x509.pem"
        echo "  $AUTH"
        exit
fi

openssl genrsa -3 -out $1.pem 2048

openssl req -new -x509 -key $1.pem -out $1.x509.pem -days 10000 \
    -subj "$AUTH"

echo "Please enter the password for this key:"
openssl pkcs8 -in $1.pem -topk8 -outform DER -out $1.pk8 -passout stdin
```

mkkey.sh is a helper script used to generate the platform keys.

The password that you enter will be displayed in your terminal window. You will need the password to sign release builds.

To generate the required four platform keys, run mkkey.sh four times, specifying the key name and password for each:

```
$sh mkkey.sh platform # enter password
$sh mkkey.sh media # enter password
$sh mkkey.sh shared # enter password
$sh mkkey.sh release # enter password
```

You should now have new keys for your product.

Signing a build for release

To sign a build for a release, perform the following steps:

1. Sign all the individual parts of the build.
2. Put the parts back together into image files.

Signing applications

Use build/tools/releasetools/sign_target_files_apks to sign a target_files (will have all files, system and recovery.img boot.img) package. The target_files package is not built by default, you need to specify the "dist" target when you call "make". For example:

```
make -j4 PRODUCT-<product_name>-user dist
```

This command above creates a file under out/dist called <product_name>-target_files.zip. This is the file you need to pass to the sign_target_files_apks script.

You would typically run the script like this:

```
./build/tools/releasetools/sign_target_files_apks -d vendor/<vendor_name>/security/
<product_name> <product_name>-target_files.zip signed-target-files.zip
```

If you have prebuilt and pre-signed apks in your build that you do not want re-signed, you must ignore them by adding `-e Foo.apk=` to the command line for each apk you wish to ignore.

sign_target_files_apks also has many other options that could be useful for signing release builds. Run it with `-h` as the only option to see the full help.

Creating image files

Once you have signed-target-files.zip, create the images so that you can put it onto a device with the command below:

```
build/tools/releasetools/img_from_target_files signed-target-files.zip signed-img.zip
```

signed-img.zip contains all the .img files. You can use fastboot in fastboot update signed-img.zip to get them on the device.

36 How do I generate a different OTA package?

Assuming you can build the OTA package sabresd_6dq_target_files-eng.xxx.zip by following the steps in "How do I build an OTA package?", this file is a full package of your current build image. Save this file in a directory. For example:

```
mkdir ~/release-1
cp out/dist/imx6q_sabrelite_target_files-eng.xxx.zip ~/release-1
```

If you find a bug or if you want to generate a new release package later, do the same operation to make a "dist" build.

Save it to a new place, for example, ~/release-2

```
mkdir ~/release-2
cp out/dist/imx6q_sabrelite_target_files-eng.xxx.zip ~/release2
```

To generate a different package, carry out the following command:

```
cd mydroid; # you must in your android root dir to do this command
./build/tools/releasetools/ota_from_target_files -i ~/release-1/imx6q_sabrelite_target_files-
eng.xxx.zip \
~/release-2/imx6q_sabrelite_target_files-eng.xxx.zip ~/diff-from-release-1-to-2.zip
```

~/diff-from-release-1-to-2.zip is a diff package between release 1 and release 2.

In this example, a kernel commit and a framework/base are changed, and the zip package is only 4.5 MB.

37 How do I update my device with the OTA Package?

Updating by Android fastboot

1. Run the command below on your PC:

   ```
   $ fastboot update update.zip
   ```
2. Reboot the board to recovery mode:

   ```
   u-boot> fastboot
   ```

3. Connect the USB cable.

Updating by Android recovery (manually)

Copy ota.zip or diff-ota.zip in the previous example to your SD card, and update your system to recovery. Choose menu to apply this update.zip to perform the upgrade.

Updating by Android recovery (automatically)

You can copy the update.zip to /cache dir, and then run the following command:

```
busybox cp /sdcard/update.zip /cache/
mkdir /cache/recovery/
echo "--update_package=/cache/update.zip" > /cache/recovery/command
reboot recovery
```

The recovery will automatically apply the command and install this update package.

Updating by Android framework API (OTA App)

You only need to do the following:

1. Check if an update package is available, and notify the user to do upgrade.
2. Download the update zip package to /cache/ dir.
3. Call recovery API to do the upgrade.

The following are the two APIs that can verify and install the update package:

```
import android.os.RecoverySystem;
RecoverySystem.verifyPackage();
RecoverySystem.installPackage();
```

38 How do I customize the reference OTA application?

In this release, we have added a reference application that can work as OTA, which will check the update from server, download and install the package to upgrade the software in the board without any external tools.

The application is under packages/apps/fsl_imx_demo/FSLOta/.

It is a dialog activity, and can be enabled through the **Settings > About > Additional System Update** menu.

The server side of this OTA application is a common HTTP server, such as lighttpd and apache.

The following is a "tree" command output of a sample OTA site:

```
$ tree ~/ota-site                          # root directory of the HTTP server
/home/user/ota-site/
-- sabresd_6dq                               # product name as the directory name
    |-- build.prop                       # for checking version
    -- sabresd_6dq.ota.zip               # name is product_name.ota.zip, the OTA application
will download this package, and this should be an update.zip
```

You can check the comment in the right side.

After you set up the HTTP server, you can change the /system/etc/ota.conf of server IP and server port.

The build.prop file is copied from this build's system/build.prop. The OTA application will download this file, parse it to get the build time, and compare with the build time of itself.

If the server build is newer, it will show the version information and package size, and prompt the user to upgrade the server. The user can upgrade the server by clicking the Update button.

Then you can see the download starting.

In this example, this application is now downloading the sabresd_6dq.ota.zip package.

After downloading is finished, the title will change to "Verify package". During this time, it is actually doing RecoverySystem.verifyPackage() API to verify whether the package is complete, like an MD5 checksum checking. It will also check whether the key chain in package aligns to the key chain in the device.

After the verifying is finished, it will call RecoverySystem.installPackage() API to install the package. This was generally similar to "How do I update my device with the OTA Package?" It will write a recovery command and store it into /cache/recovery/command, and reboot the system.

After reboot, the system will boot to recovery mode. After it installs the update package, you should see the “Android Robot” spinning on the screen. If you meet an error, you should see the "Error Robot", and it will stop spinning. Press the MENU button to show the log output on the screen.

39 How do I customize the update script to update uboot?

Because Android only upgrades the boot.img, system.img, and recovery partitions, the automatically generated update package does not support upgrading bootloader. If you need to upgrade the bootloader, you need to modify the update package and perform the signing work manually.

1. Unzip the update.zip, and then modify the updater_script by implementing the following operations.

   To upgrade uboot to NOR flash, refer to this script:

   ```
   ui_print("writting u-boot...");
   write_raw_image("u-boot.bin", "/dev/mtd0");
   show_progress(0.1, 5);
   ```

   To upgrade uboot for eMMC storage, because u-boot may be stored in the "boot partition" of eMMC, you need to perform some system file operations before dd, for example,

   ```
   # Write u-boot to 1K position.
   # u-boot binary should be a no padding uboot!
   # For eMMC(iNand) device, needs to unlock boot partition.
   ui_print("writting u-boot...");
   package_extract_file("files/u-boot-no-padding.bin", "/tmp/u-boot-no-padding.bin");
   sysfs_file_write("class/mmc_host/mmc0/mmc0:0001/boot_config", "1");
   simple_dd("/tmp/u-boot-no-padding.bin", "/dev/block/mmcblk0", 1024);
   sysfs_file_write("class/mmc_host/mmc0/mmc0:0001/boot_config", "8");
   show_progress(0.1, 5);
   ```

2. Resign the update package by using the following command:

   ```
   $ make_update_zip.sh ~/mydroid ~/update-dir
   ```

40 How do I make a fake battery and charger status report to some applications?

There is no battery and charger in the Freescale i.MX6DQ/DL SABRE-AI board and i.MX6SoloLite EVK board. The pre-condition of certain features or functions of specific application is the battery or charger status, such as data partition encryption features in the Setting application. Taking i.MX6SoloLite EVK board as an example, you can make the system to report a fake battery and charger status to enable such certain features or functions as below. It will make the system show in fake 100% level of battery and AC charger plugged in.

```
diff --git a/evk_6sl/init.rc b/evk_6sl/init.rc
index 934828a..82f9c3f 100644
--- a/evk_6sl/init.rc
+++ b/evk_6sl/init.rc
@@ -19,6 +19,9 @@ on init

on boot

+    # emmulate battery property
+    setprop sys.emulated.battery 1
+
     # Set permission for IIM node
     symlink /dev/mxs_viim /dev/mxc_mem
```

41 How to enable consumer IR in android KitKat?

Android 4.4 KitKat supports the consumer IR feature. To support this feature, PAD_DISP0_DAT9 should be configured as PWM2 output in Freescale i.MX 6Dual/Quad and i.MX 6DualLite SABRE-SD board. The sample schematic of IR transmitter is as follows:

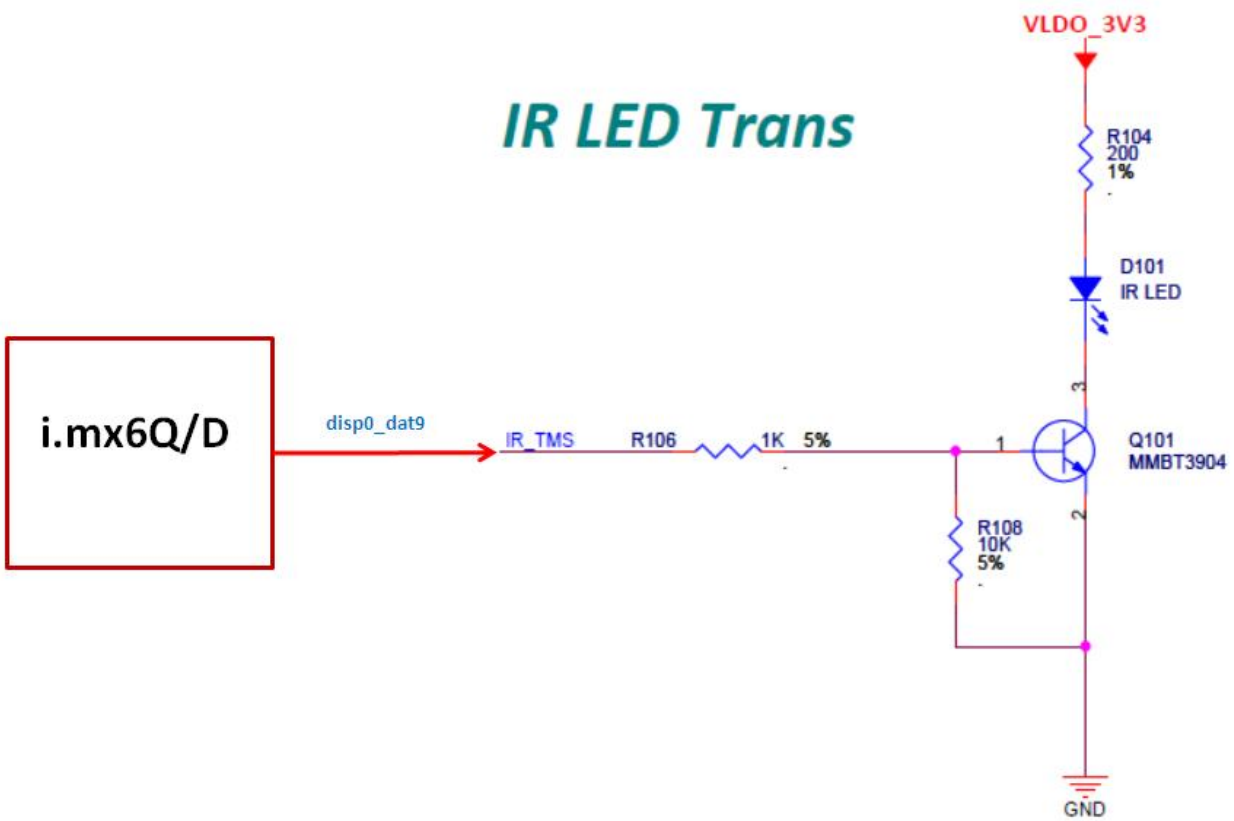

Figure 3. Sample schematic of IR transmitter

EPIT timer 0 and PWM2 are used to generate and control the IR transmit signal. A new i.MX 6 IR driver is introduced. The EPIT driver, PWM driver, and IR related driver should be enabled in kernel configuration file, such as the imx6_android_defconfig file list below.

```
diff --git a/arch/arm/configs/imx6_android_defconfig b/arch/arm/configs/
```

```
imx6_android_defconfig
            @@ -346,12 +347,16 @@
            CONFIG_USB_EHCI_ARC_H1=y
            CONFIG_USB_FSL_ARC_OTG=y
            # CONFIG_MX6_INTER_LDO_BYPASS is not set  CONFIG_MX6_CLK_FOR_BOOTUI_TRANS=y
            +CONFIG_MX6_IR=y
            CONFIG_ISP1504_MXC=y
            # CONFIG_MXC_IRQ_PRIOR is not set
            CONFIG_MXC_PWM=y
            +CONFIG_MXC_EPIT=y
            # CONFIG_MXC_DEBUG_BOARD is not set
            +CONFIG_HAVE_EPIT=y
            CONFIG_MXC_REBOOT_MFGMODE=y
            CONFIG_MXC_REBOOT_ANDROID_CMD=y
            +# CONFIG_MXC_USE_EPIT is not set
            CONFIG_ARCH_MXC_IOMUX_V3=y
            CONFIG_ARCH_MXC_AUDMUX_V2=y
            CONFIG_IRAM_ALLOC=y
            @@ -1605,6 +1610,7 @@
            CONFIG_SERIAL_CORE_CONSOLE=y
            # CONFIG_SERIAL_PCH_UART is not set
            # CONFIG_SERIAL_XILINX_PS_UART is not set
            # CONFIG_TTY_PRINTK is not set
            +CONFIG_DEVIR=y
            CONFIG_FSL_OTP=y
```

The consumer IR feature can be tested by ApiDemo or CTS.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AFAQ
Rev. kk4.4.2_1.0.0-ga
07/2014