

# **i.MX Android**

## **Advanced User Guide**

R14 Beta  
June 11, 2012

[i.MX Android Advanced User Guide](#)

[1 Overview](#)

[2 Output display customization](#)

[2.1 24bpp/16bpp pixel display panel](#)

[3 Camera & Video Recorder](#)

[3.1 How to change the default setting](#)

[3.1.1 Change the default preview size](#)

[3.1.2 Change the default video recorder setting](#)

[4 Memory Layout](#)

[5. Storage Customization](#)

[6. Customization on the boot logo](#)

[7. Connectivity customization](#)

[7.1 About GPS part](#)

[7.2 About WIFI part](#)

[7.3 3G connection](#)

[7.3.1 Device Port and Data Port](#)

[7.3.2 APN setting](#)

[8. Customization MFGTool](#)

[9. Customization of the USB gadget's Product ID and Vendor ID](#)

[Change the VID/PID in init.usb.rc](#)

[Update the HOST's configuration](#)

[For Windows PC:](#)

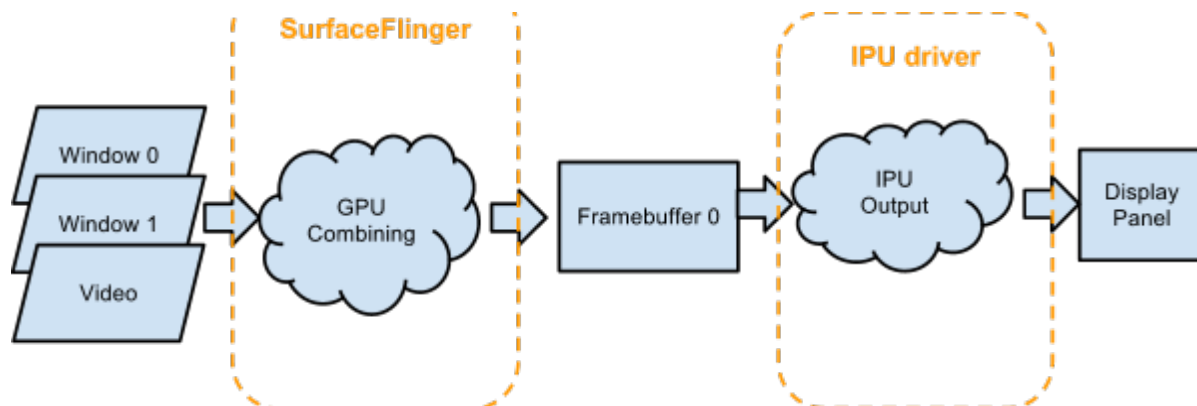
[For Linux PC :](#)

# 1 Overview

This document is supposed to provide advanced information as how to customize or extend the feature lists in the R14 Beta release package. It is primarily target to android framework and kernel developer who want to customize their product based on R14 Beta.

## 2 Output display customization

The default image including in this package use the LVDS as the primary output with 16bpp enabled. The diagram below demos the typical data path for each UI component(Windows and video) to the output display panel.



The description and data format for each node in the data path as describe as below:

- Window0-n:** each window represents a UI surface in the framework. The application draws their UI data into its window. The data format can be RGB565, or RGBA8888. There are two factors to impact the data format on the windows.  
 If Application request transparent feature in its UI component, the data format is RGBA8888.  
 If no alpha blending feature requested, the data format is decided by the frame buffer 0' format. If the frame buffer 0 is 16bpp(RGB565) format, the non-transparent windows format is RGB565. If the frameb uffer 0 is 32bpp(RGBA8888) format, the non-transparaent windows format is RGBA8888.

Transparent Request	Frame buffer 0' format	Windows' format
Y	16bpp	RGBA8888
Y	32bpp	RGBA8888
N	16bpp	RGB565
N	32bpp	RGBA8888

- Video/Camera:** The video playback or camera preview surface is created for display of decoded picture buffers or camera preview respectively. In most cases, it should be YUV format(NV12 or YV12) and the number of buffers depends on the user case. For video playback, it may vary from 6~20 for different video codec's profile, and for camera, the buffer count is been set to 5.

## 2.1 24bpp/16bpp pixel display panel

Mostly there are two kinds of output display panel, 24bit or 16bit. For a 16bit display panel, the digital format of the frame buffer 0's pixel data is RGB565. The bpp setting can be configured in U-Boot by defining bpp=16(for 16bits) or bpp=32(for 24bits) in the kernel command line arguments. depending on your output display. If no bpp setting is defined, the default bpp is 16. An example of the kernel argument is below:

```
video=mxcdi0fb:RGB666,LDB-XGA,bpp=32
```

For a 24bit display panel, the digital format of the frame buffer 0's pixel data can be RGB565 or RGBA8888. The below log output from "logcat" during system bootup can be used to indicate which pixel format is set in the frame buffer:

```
W/imx5x.gralloc( 2253): 32bpp setting of Framebuffer caught!
W/imx5x.gralloc( 2253): using (fd=12)
W/imx5x.gralloc( 2253): id          = DISP4 BG - DI1
W/imx5x.gralloc( 2253): xres       = 1024 px
W/imx5x.gralloc( 2253): yres       = 768 px
W/imx5x.gralloc( 2253): xres_virtual = 1024 px
W/imx5x.gralloc( 2253): yres_virtual = 2304 px
W/imx5x.gralloc( 2253): bpp      = 32
W/imx5x.gralloc( 2253): r          = 16:8
W/imx5x.gralloc( 2253): g          = 8:8
W/imx5x.gralloc( 2253): b          = 0:8
W/imx5x.gralloc( 2253): width      = 163 mm (159.568100 dpi)
W/imx5x.gralloc( 2253): height     = 122 mm (159.895081 dpi)
```

The amount of memory which needs to be reserved for frame buffer 0 depends on your type of output display panel and the digital format of the pixel data. To reserve this memory, the kernel parameter "fbmem=nM" is used. An example on how to derive the "n" portion of this argument is below:

For a 24bpp output display panel, the chosen frame buffer 0's pixel data format is RGBA8888, which equates to 32bpp. The formula and definition of the fields to calculate "n" is below:

$$n = width * height * buffer[\backslash,]number * sizeof(format)$$

width	The width of the display panel in pixels.
height	The height of the display panel in pixels.
buffer number	The number of buffers for frame buffer 0. The default value is 3.
sizeof(format)	The size of each pixel in bytes. For 16bpp, sizeof(format) is 2 bytes and for 32bpp, the sizeof(32bpp) is 4 bytes.

The memory allocation for each UI, Video, and Camera surface is also dependent upon the pixel data formats used for the surfaces. The amount of memory which needs to be reserved for these surfaces is difficult to predict since different use cases require different surfaces to be created in the system. For more information on the kernel parameters used to reserve the memory for these surfaces, please see Section 4.

## 3 Camera & Video Recorder

The OV5642 is the camera sensor supported on the i.MX53 SMD. In this default image, the below capability is included in the Camera & Video Recorder feature through the default Android Camera Application:

- The default preview size is 640x480
- The default video recorder has three profile options:

Profile	Resolution	Video Codec	Frame Rate	Audio Codec	Audio Sample Rate	Output File Format
720P(HD)	1280x720	H.264	30	MP3	44.1KHZ	MP4
480P(SD)	640x480	H.263	30	AMR-NB	8KHZ	3GP

### 3.1 How to change the default setting

#### 3.1.1 Change the default preview size

Change the bold code in myandroid/hardware/imx/mx5x/libcamera/CameraHal.cpp  
CameraHal :: InitCameraBaseParam(). Make sure the changed preview size been supported by the camera sensor

```
pParam->set(CameraParameters::KEY_SUPPORTED_PICTURE_SIZES,
mSupportedPictureSizes);
pParam->set(CameraParameters::KEY_SUPPORTED_PREVIEW_SIZES,
mSupportedPreviewSizes);
pParam->set(CameraParameters::KEY_SUPPORTED_PREVIEW_FRAME_RATES,
mSupportedFPS);
pParam-
>set(CameraParameters::KEY_SUPPORTED_PREVIEW_FPS_RANGE, "(15000,15000),(30
000,30000)");
pParam->set(CameraParameters::KEY_PREVIEW_FPS_RANGE, "30000,30000");

pParam->setPreviewSize(640, 480); //Change the default preview size
here
```

### 3.1.2 Change the default video recorder setting

The video recorder configure file is in device/fsl/proprietary/omx/registry/media\_profiles\_720p.xml, in which the Camera id, recorded video properties can be configured. Change in device/fsl/proprietary/omx/fsl-omx.mk can make a different Camcorder configure.

```
ifeq ($(BOARD_SOC_TYPE),IMX51)
LOCAL_SRC_FILES := registry/media_profiles_d1.xml
endif

ifeq ($(BOARD_SOC_TYPE),IMX53)
LOCAL_SRC_FILES := registry/media_profiles_720p.xml
endif

#for mx6x, it should be up to 1080p profile
ifeq ($(BOARD_SOC_TYPE),IMX6Q)
LOCAL_SRC_FILES := registry/media_profiles_1080p.xml
endif
```

#### Notes:

- For **cameraId**, 0 is mapped to the front camera if it is available.
- For default, only one Camera for video recording is enabled. **To add another camera**, you can add another **CamcorderProfiles** item according to the first one.
- For video size, it should be the set that the sensor supports.
- For bitrate, for VGA recording, bitrate is recommended to be 1500000, for 720p recording, bitrate is recommended to be 3000000.
- Different sensor should have different different media\_profiles.xml setting due to the capability limitation of camera sensor.
  - ov5642 sensor can have the Camcorder configure up to media\_profiles\_1080p.xml.
  - ov3640 sensor can only have the Camcorder configure up to media\_profiles\_d1.xml.

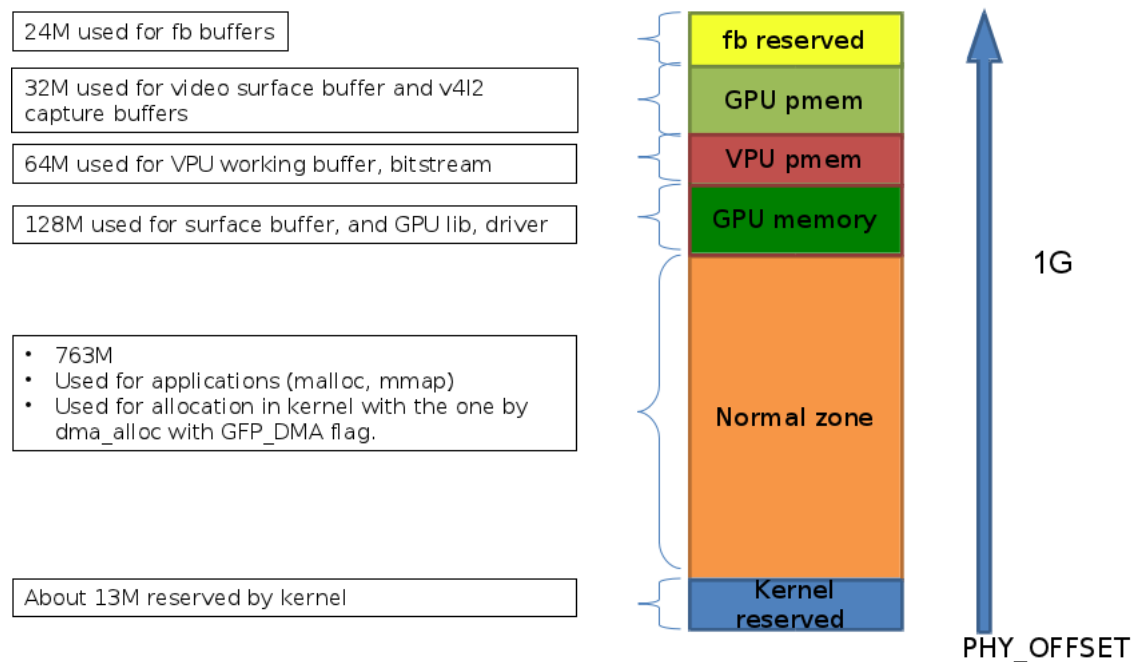
## 4 Memory Layout

This section describes the i.MX Android memory usage and layout for the total system. For i.MX Android, the DDR memory has been divided into 6 regions. From bottom to top they are:

- Linux Kernel reserved space

- Normal zone space managed by kernel's MM (high memory zone is also included)
- Reserved memory for GPU libs, drivers and normal surface double buffers
- Reserved pmem for VPU's working buffer and bitstream
- Reserved pmem for GPU
- Reserved space for frame buffer BG/FG triple buffers

The following diagram shows the default memory usage and layout on i.MX platform.



Default memory usage on i.MX53 platform

The kernel reserved region size is controlled by the kernel and cannot be adjusted via kernel parameters, and the Normal zone region depends on the total DDR size and the reserved spaces. The reserved GPU memory size is configurable via the kernel parameter "gpu\_mem=nM", although it is recommended to leave the value the default.

For the other regions, the customer can adjust the reserved memory size for BG(background) frame buffer and GPU/VPU pmem, and the sizes depend on their use case. The kernel parameters for to control the reserved amounts are below:

- for frame buffer  
fbmem=<fb0 size>,<fb1 size>
- for pmem  
pmem=<gpu size>,<vpu size>

For example:

You have two display devices, one is XGA LVDS, the other is HDMI 1080P device (default 32bpp). Then you have to specify the BG buffer size for them:



```
fbmem=32M,64M
```

```
fbmem=10M,24M
```

where 10M  $\sim$  1024x768x4(32bpp)x3(triple buffer)

24M  $\sim$  1920x1080x4(32bpp)x3(triple buffer)

The GPU pmem and VPU pmem can be configured through the below parameters in U-Boot:

```
pmem=128M,64M
```

The first size "128M" is for the GPU pmem and the second size "64M" is for the VPU pmem. If no specific parameters setting for pmem in U-Boot, the kernel default setting is "pmem=128M,64M". The VPU pmem is for internal buffers used during the video decode process. The size may need to increase if multiple video codec instances are used at the same time, like for Video conferencing.

## 5. Storage Customization

There are multiple devices supported in the system:

- TF card
- SD card in external MMC/SD slot
- UDISK connected to the board through the USB port

These are mounted to /sdcard, /extsd and /udisk in Android root file system respectively, and all these can be accessed through MTP by PC host.

To support more disks, please change the following files according to the sample:

- device/fsl/imx53\_smd/vold.fstab, or the path according to the board used, for example:

```
dev_mount extsd /mnt/extsd auto /devices/platform/sdhci-esdhc-imx.2/  
mmc_host/mmc1
```

- device/fsl/imx53\_smd/overlay/framework/base/core/res/res/xml/storage\_list.xml,

for example:

```
<storage android:mountPoint="/mnt/extsd"  
    android:storageDescription="@string/storage_usb"  
    android:removable="true"  
    android:primary="false" />
```

## 6. Customization on the boot logo

If there is a file called initlogo.rle under the root folder, Android displays it while booting. Complete following steps to create a RLE compressed picture from a PNG file.

1. Find an image you like
2. Edit it with your favorite editing suite and scale it to 1024x768
3. Save it as a PNG without alpha channel/transparency.
4. Use the convert tool from the ImageMagick toolkit: `convert -depth 8 splash.png rgb:splash.raw`
5. Use the Android tool in `mydroid/out/host/linux-x86/bin/rgb2565`(this tool gets built when we do full system builds or build it in `mydroid/build/tools/rgb2565: gcc -o rgb2565 to565.c`)
6. Run the conversion command: `rgb2565 -rle < splash.raw > splash.rle`
7. Copy `splash.rle` to `mydroid/device/fsl/imx5x/initlogo.rle`. Do a full system build to generate `ramdisk.img`.

## 7. Connectivity customization

This section describes how to port connectivity part( including GPS, WIFI and 3G) on i.MX53 SMD board.

### 7.1 About GPS part

This release supports GPS module AH-1613 from LOCOSYS, and it's a module AR1520A GPS chip from Atheros. The UART2 is used to communicate with GPS module. In addition, GPS\_REST\_B need to customized for GPS reset. The GPS software version is OrionM4.2. The Orion package includes the following bin files which will provide Orion service and IPC server for GPS:

- ingsvcd
- OrionSys.so
- libOrionCtl.so
- AR1520A-Auto.img
- Orion.ini

How to port hardware abstraction layer between the GPS hardware and Android Location Manager:

1. athr\_gps.c
2. Android.mk  
Modifications in vendor BoardConfig.mk(mydroid/device/fsl/imx53\_smd):
1. BOARD\_HAVE\_HARDWARE\_GPS := true
2. USE\_ATHR\_GPS\_HARDWARE := true
3. USE\_QEMU\_GPS\_HARDWARE := false  
Modification in vendor init.rc(mydroid/device/fsl/imx53\_smd):

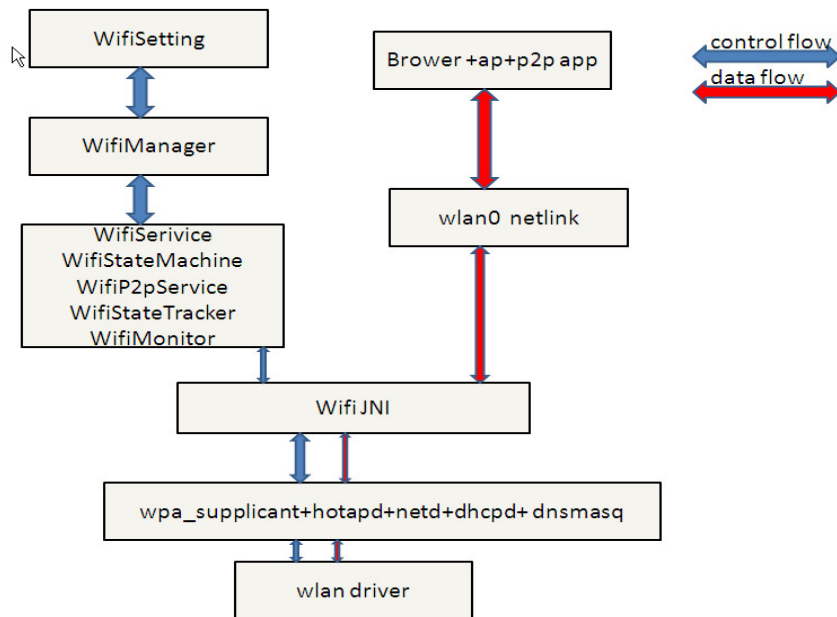
```
#Set GPS serial and reset GPIO pin
symlink /dev/ttymxc2 /dev/gpsdevice
setprop ro.kernel.android.gps ttya
write /sys/class/gpio/export 60
write /sys/class/gpio/gpio60/direction "out"
write /sys/class/gpio/gpio60/value 1
# Orion InG
service ing /system/etc/gps/ingsvcd -c /system/etc/gps/Orion.ini
    user root
    group gps
    disabled
    oneshot
```

For GPS power, it is configured in kernel/arch/arm/mach-mx5/board-mx53\_smd.c. For GPS debug log, you can look up the Orion.log located in /data/app folder.

## 7.2 About WIFI part

Atheros AR6103 is the default WIFI module supported in this package.

WIFI code topology as below:



#### 1. WIFI control flow

The WIFI setting app supports the control of STA, AP and P2P profiles. You can customize it through modifying the following values in `required_hardware.xml` which is located at `myandroid/device/fsl/imx53_smd`:

```
<feature name="android.hardware.wifi" />
<feature name="android.hardware.wifi.direct" />
```

The WIFI app will realize enable/disable, scan, discover, connect/disconnect calls through WIFI services in framework. For this part, no changes are needed but only use the google's code. Through JNI, WIFI services will communicate with `wpa_supplicant` or `hostapd`.

In STA and P2P case, `wpa_supplicant` plays an important part in making a bridge between WIFI framework and driver. Due to the default `wpa_supplicant` from ASOP is based on Broadcom wifi driver, you have to make some changes for other vendor's WIFI module.

#### 2. How to change WIFI MAC?

For Atheros wifi, you can enable the macro `ATH_SOFTMAC_FILE_USED` in `AR6KSDK.build_3.1_RC.563/host/localmake.linux.inc` file to let it read MAC address from "softmac" file which is located in `/system/wifi/` folder. In softmac file, you can customize a certain MAC address like `02:03:7F:07:84:22`. If there is no softmac file in `/system/wifi/` folder, it will generate a random MAC address which will be written into EEPROM of WIFI.

module's firmware. In this way, the WIFI MAC address will be changed every time when disabling/enabling WIFI.

You can follow the following steps to customize a certain fix MAC address:

```
mount -w -o remount /dev/block/mmcblk0p5 /system
cd /system/wifi
busybox vi softmac
```

### 3. How to use p2p profile?

In order to use p2p, you will have to ensure two peers have different MAC addresses. Because UUID (universally unique identifier) is unique for a device, so this parameter `inwpa_supplicant_p2p.conf` should also differ for each other. Or connect process will report error. This will be optimized in later software version. For different peers, they will form a workgroup with one peer acting as owner and others as clients. About who will act as group owner, it is determined by `p2p_go_intent` parameter in `wpa_supplicant_p2p.conf`.

## 7.3 3G connection

3G data connection has been enabled in this release. The 3G modules been supported are Huawei EM770W and Infinion Amazon1. The 3G modem is connected with i.MX hardware through miniPCIe interface.

### **Huawei EM770W**

The Huawei EM770W will be present in android system as several ttyUSB device as below:

```
/dev/ttyUSB0
/dev/ttyUSB1
/dev/ttyUSB2
/dev/ttyUSB3
/dev/ttyUSB4
```

### **Infinion Amazon1**

The Infinion Amazon1 will be present in android system as several ACM device as below:

```
/dev/ttyACM0
/dev/ttyACM1
```

```
/dev/ttyACM2  
/dev/ttyACM3  
/dev/ttyACM4  
/dev/ttyACM5  
/dev/ttyACM6
```

### 7.3.1 Device Port and Data Port

For each 3G modem, there are two devices will be used in Android telephony framework(also known as RIL framework). One is called device port. The other is called data port. The device port is used to send AT commands or receive the AT reponses from 3G modem. The data port is used to set up the ppp data connection, on which the 3G data connection is based. For Huawei 3G EM770W 3G modem, "/dev/ttyUSB2" is the device port, and "/dev/ttyUSB0" is the data port. For Infinion Amazon1, "/dev/ttyACM3" is the device port, and the "/dev/ttyACM0" is the data port. To support other 3G modem, please consult the modem's specification to get the right device port and data port. And add the 3G modem support as below change in *myandroid/hardware/ril/reference-ril/runtime\_port.c*.

```
static struct modem_3g_device modem_3g_device_table[] = {  
    {  
        .name      = "Huawei-EM770",  
        .idVendor  = "12d1",  
        .idProduct = "1001",  
        .deviceport = "/dev/ttyUSB0",  
        .dataport  = "/dev/ttyUSB0",  
        .type      = HUAWEI_MODEM,  
    },  
    ...  
    {  
        .name      = "xxx", //3G modem's name, should be unique in this table  
        .idVendor  = "xxxx", //USB vendor id of the 3G modem  
        .idProduct = "xxxx", //USB product id of the 3G modem  
        .deviceport = "/dev/xxxn", //Device port of the 3G modem  
        .dataport  = "/dev/xxxm", //Data port of the 3G modem  
        .type      = MODEM_TYPE, //Modem type, which you can add specific AT  
        commands based the modem type.  
    }  
};
```

### 7.3.2 APN setting

Different mobile operator should have different APN setting. The default APN setting enabled is based on the mcc&mnc number of the mobile operator. Android telephony framework will find the one apn setting in myandroid/device/fsl/apns-conf.xml which has the same mcc&mnc number. To make the APN automatically enabled, the right APN setting entry should be added into myandroid/device/fsl/apns-conf.xml as below:

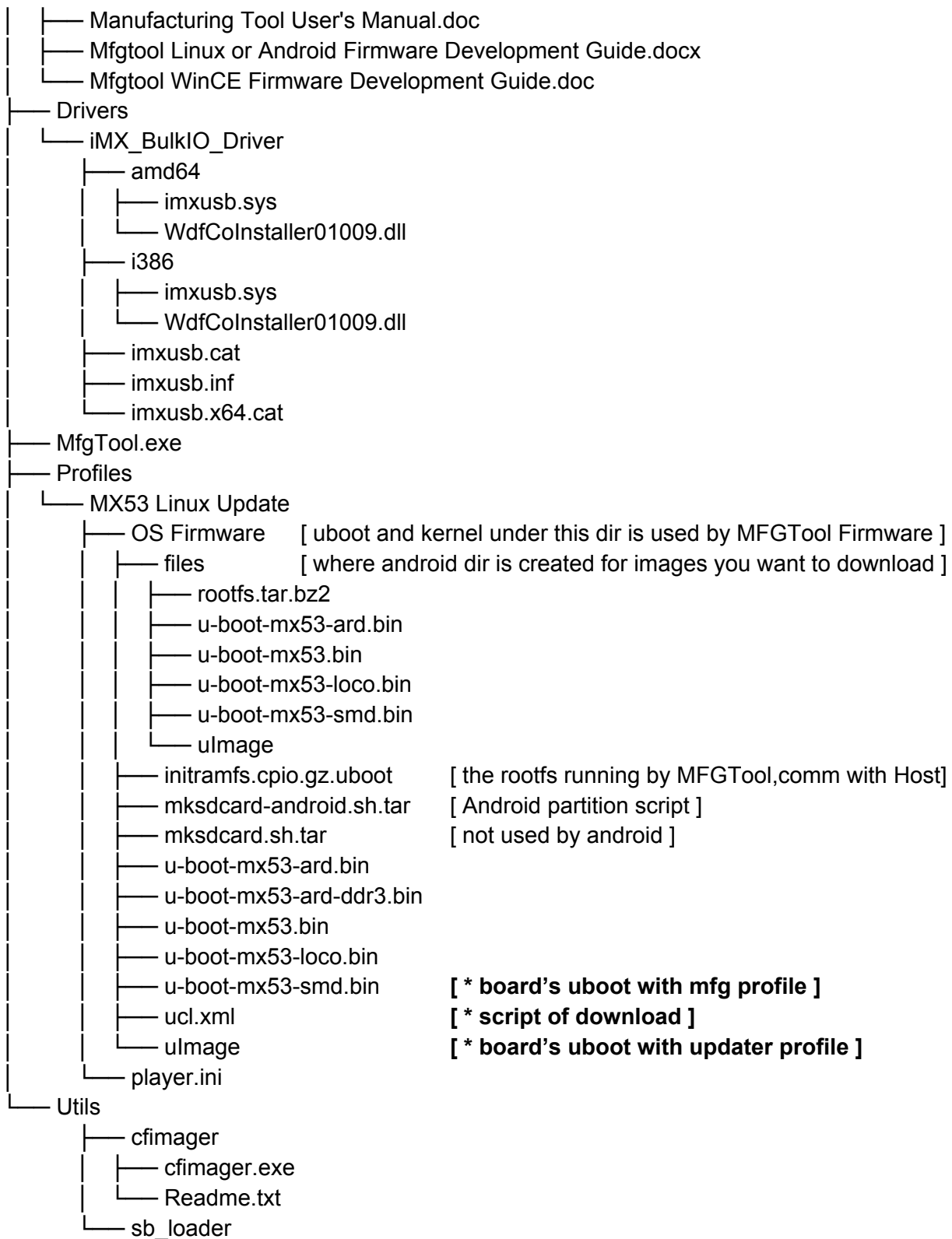
```
<apns version="7">
  <apn carrier="T-Mobile US"
    mcc="310"
    mnc="260"
    apn="epc.tmobile.com"
    user="none"
    server="*"
    password="none"
    mmsc="http://mms.msg.eng.t-mobile.com/mms/wapenc"
  />
  ...
  <apn carrier="xxxx xxxx" //Mobile Operator name
    mcc="xxx" //mcc number of mobile operator
    mnc="xx" //mnc number of mobile operator
    apn="xxxxxx" //apn setting for data connection
    user=""
    server=""
    password=""
    mmsc=""
  />
</apns>
```

## 8. Customization MFGTool

The MFGTool is a tool can be used to download firmware into i.MX board. There is a user guide about how to use this tool in "Quick Start" and "User Guide" document.

Below is the description of the MFGTool directory tree(output of 'tree' command with comments):

```
|— Document
|   |— Manufacturing Tool Factory Operation manual.docx
|   |— Manufacturing Tool Quick Start Guide.doc
|   |— Manufacturing Tool release note.docx
|   |— Manufacturing Tool UCL user manual.doc
```





└─ sb\_loader.exe  
└─ UserGuide.txt

After you get a view of mfgtool 's directory, Let's have a example of how to customize a MFGTool firmware (Board is i.MX53 SMD in this example, change the name of U-Boot per your needs)

First, you will notice these images

- **Profiles/MX53 Linux Update/OS Firmware/u-boot-mx53-smd.bin**
- **Profiles/MX53 Linux Update/OS Firmware/ulmage**

You need to replace these two images if you want to download images into your board. The U-Boot and kernel images are **NOT** the images you want to download into board, they are tool firmware images used to communicate with MFGTool and download images into your board. The images you want to download is under **Profiles/MX53 Linux Update/OS Firmware/files/android/** directory.

For special U-Boot, you need to build boot loader with mfg profile:  
eg:

```
make distclean  
make mx53_smd_mfg_config
```

For special kernel image, you need to build kernel with below config:

```
make imx5_updater_defconfig  
make ulmage -j4
```

Then replace these two output images with original images.

**Note:**

You must make sure you have enable your BOARD in kernel and U-Boot's configure file.

## 9. Customization of the USB gadget's Product ID and Vendor ID

In ICS release, we use Google's VendorID and ProductID for USB gadget functions. This can make it easier on the customer using ADB by supporting the default Google windows driver in Android SDK. If you want to change the ID to your own one, please do the

following steps:

### 1. Change the VID/PID in init.usb.rc

Android ICS exports the USB gadget functions VID/PID configuration in the Android gadget driver by sysfs, and user can configure them in the init.usb.rc. So you need to change all the VID/PID in the init.usb.rc file to let system set the correct VID/PID when doing gadget functions'. For example: change all the /sys/class/android\_usb/android0/idVendor, and idProduct to your VID/PID in the **device/fsl/imx53\_smd/init.freescale.usb.rc**:

```
# USB massstorage configuration, with adb
on property:sys.usb.config=mass_storage,adb
    write /sys/class/android_usb/android0/enable 0
    write /sys/class/android_usb/android0/idVendor <your vid>
    write /sys/class/android_usb/android0/idProduct <your pid>
    write /sys/class/android_usb/android0/functions $sys.usb.config
    write /sys/class/android_usb/android0/enable 1
    start abdd
    setprop sys.usb.state $sys.usb.config
# USB massstorage configuration
on property:sys.usb.config=mass_storage
    write /sys/class/android_usb/android0/enable 0
    write /sys/class/android_usb/android0/idVendor <your vid>
    write /sys/class/android_usb/android0/idProduct <your pid>
    write /sys/class/android_usb/android0/functions $sys.usb.config
    write /sys/class/android_usb/android0/enable 1
    setprop sys.usb.state $sys.usb.config
.....
```

### 2. Update the HOST's configuration

This configuration change is mainly for ADB function. The MTP/PTP function does not need any changes on HOST side. The RNDIS needs to update the VID/PID in the tool/tetherxp.inf file from the release package.

#### For Windows PC:

- Download the Android SDK.
- Update the adb configuration to scan for your pid:
  - Run the SDK's tools to generate a configure file:  
android-sdk-windows\tools\android.bat update adb
  - Modify the files:  
X:\Profile\<your account>\.android\adb\_usb.ini, to add your vendor id,

e.g: 0x15a2:

```
# ANDROID 3RD PARTY USB VENDOR ID LIST -- DO NOT EDIT.  
# USE 'android update adb' TO GENERATE.  
# 1 USB VENDOR ID PER LINE.  
0x15a2
```

- Restart the adb server
  - adb kill-server
  - adb start-server

#### **For Linux PC :**

- Download the Android SDK.
- Update the adb configuration to scan for your pid:
  - Run the SDK's tools to generate a configure file:  
android-sdk-linux\_86/tools/android update adb
  - Modify the files: ~/.android/adb\_usb.ini, to add your vendor id, e.g.  
0x15a2:

```
# ANDROID 3RD PARTY USB VENDOR ID LIST -- DO NOT EDIT.  
# USE 'android update adb' TO GENERATE.  
# 1 USB VENDOR ID PER LINE.  
0x15a2
```

- Create a new udev rule file under the PC's /etc/udev/rules.d/ named: imx-android.rules. And fill in the following line into the file:

```
SUBSYSTEM=="usb", SYSFS{idVendor}=="15a2", MODE="0666"
```

- Change the new udev rule file's permission:  
chmod a+r /etc/udev/rules.d/imx-android.rules
- Connect the Android Device
- Restart the adb server
  - adb kill-server
  - adb start-server