深圳葡萄雨技术有限公司	文档版本	密级
	所属范围	共　页

葡萄雨技术 M9+Q 开发板使用手册

拟制:		日期:	
Prepared by	_____	Date	_____
审核:		日期:	
Reviewed by	_____	Date	_____
审核:		日期:	
Reviewed by	_____	Date	_____
批准:		日期:	
Granted by	_____	Date	_____

深圳葡萄雨技术有限公司

版本历史

下表指明了目前版本继以往版本的具体修改点，以示区分。下表列出了本文档全部版本历史以及修改点。

版本	日期	提出人	描述
A	Jan 2016	刘强	首次发放

目录

1 简介

1.1 文档概述

本文档作为M9开发板的开发指导。该文档介绍了使用M9开发板进行开发所需的必要操作。

1.2 原理框图

M9+Q 开发板功能原理框图请参考图1-1。

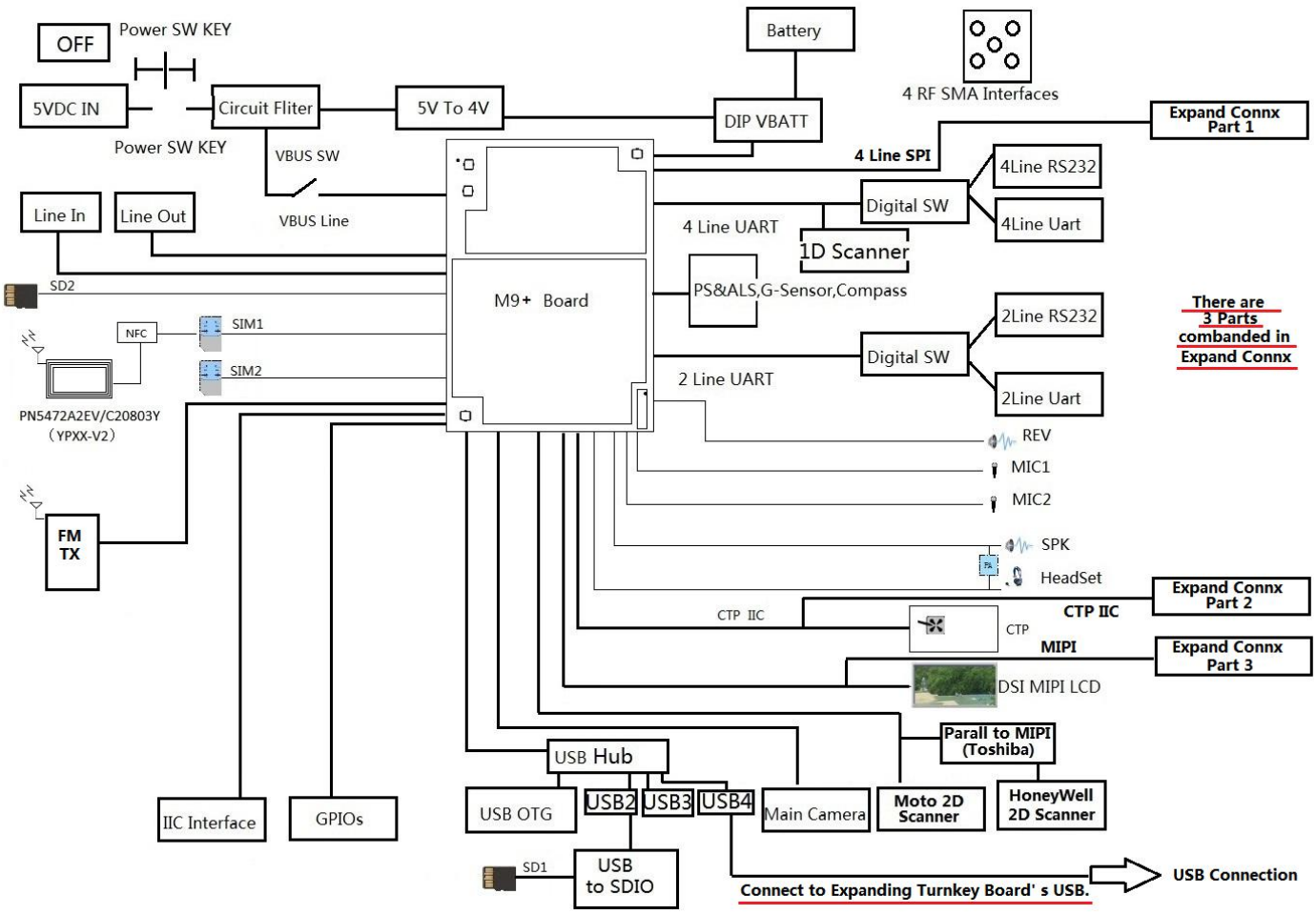
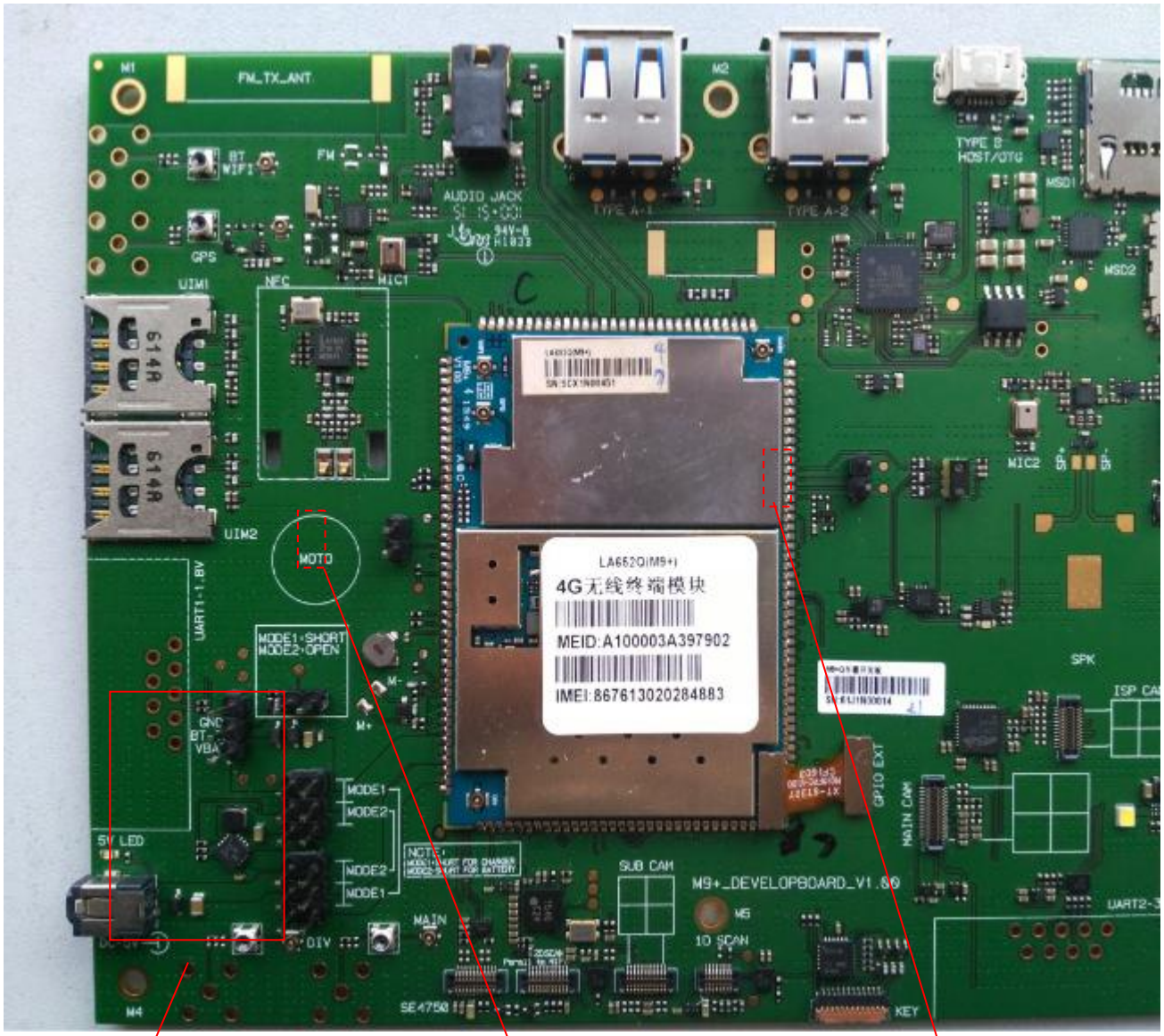


图1-1

1.2 开发板布局

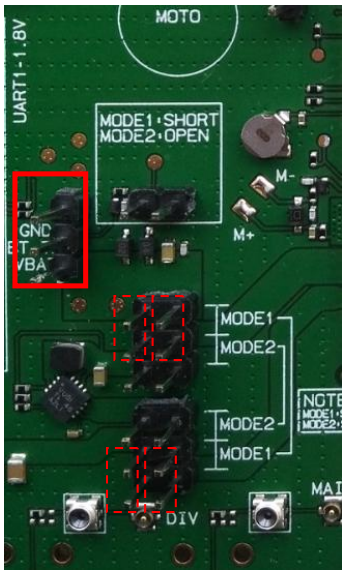
M9+Q 开发板布局图请参考图 1-2。



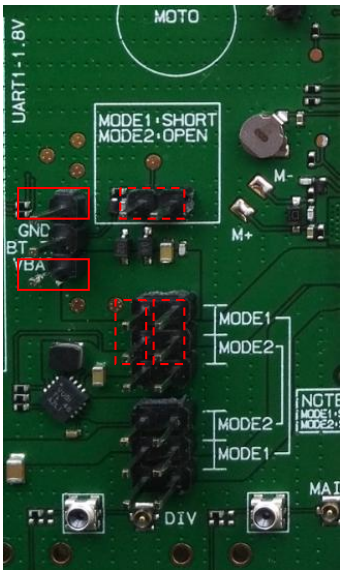
切换系统供电方式:详见下图

CBL 开机功能:短接两针可实现开机(逻辑同 Power key,可编辑)

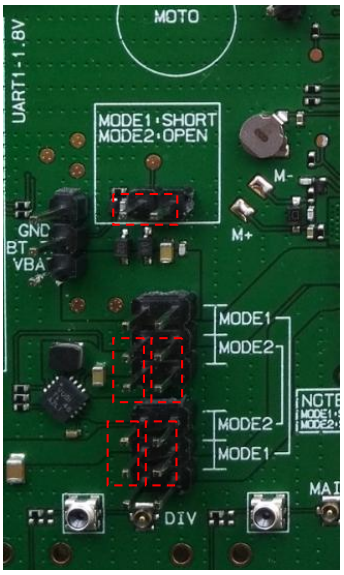
强制下载功能:短接两针,连接电脑可出强制下载口，实现强制下载



图示：电池供电;同时圆孔可给电池充电



图示：外部仪表供电（接+/-，中间脚不接);



图示：圆孔经转换后供电

图1-2

1.3 跳线功能描述

参考图示说明

2 上电及复位

2.1 上电

开发板系统开机方式主要有以下几种：

1 电源键开机（低电平时间最小持续 15.6mS（可配置））；
2 CBL 开机,同电源键；
3 充电开机；

2.2 复位

开发板提供的系统复位方式有以下几种；

1 电源键（长按 8-12S（可配置））；
2 复位键（音量下键）；
3 电源键+复位键；

3 安装USB驱动

1 使用WinRAR解压提供的USB驱动压缩包,比如CDROM_20150410A.ISO；

图3-1

2 运行压缩包中的Setup.exe，点击“安装”继续；

图3-2

3 显示如下界面后完成驱动安装。

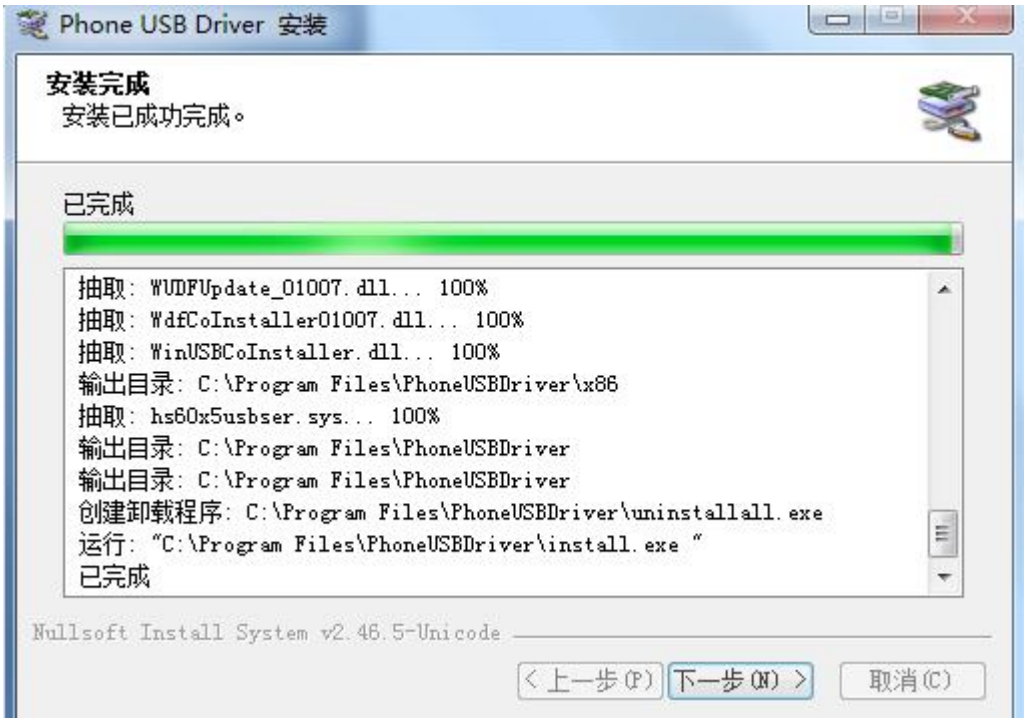


图3-3

4 烧写/下载固件

4.1 强制下载模式

进入下载模式前,请先确保开发板处于下电状态；然后将S2跳线短路，为开发板上电，再插入USB，等待电脑设备管理器中出现9008端口设备时，表明开发板已经进入强制下载模式，如图4-2。进入强制下载模式后，可使用QFIL工具进行Bin文件的烧录。

进入设备管理器步骤请参考图4-1，4-2。

图4-1

图4-2

4.2 使用QFIL工具下载Bin文件

请下载并安装新版本的QPST工具包（Version ≥ 2.7.421）。

1 开发板进入强制下载模式或开发板开机后，都可以使用QFIL进行下载。如果是强制下载模式，QFIL识别的端口为9008口；如果是正常开机模式，QFIL识别的端口是普通诊断口。

使用USB连接开发板和PC，此时QFIL会自动识别到开发板；

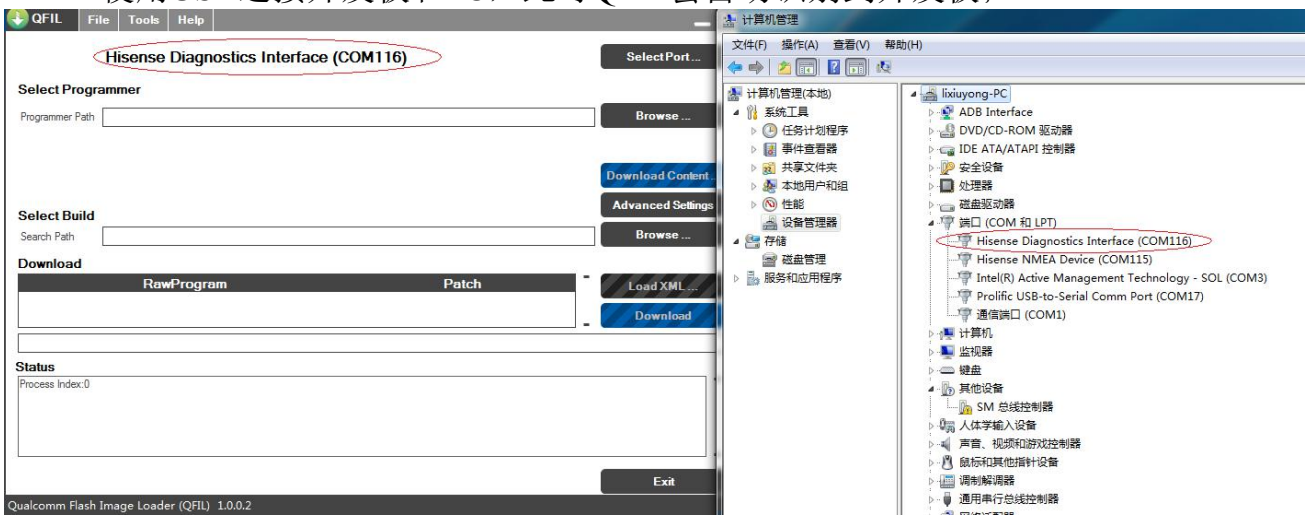


图4-3

2 点击“Browse”按钮，从Bin文件所在目录里选择8916平台的emmc_firehose文件；

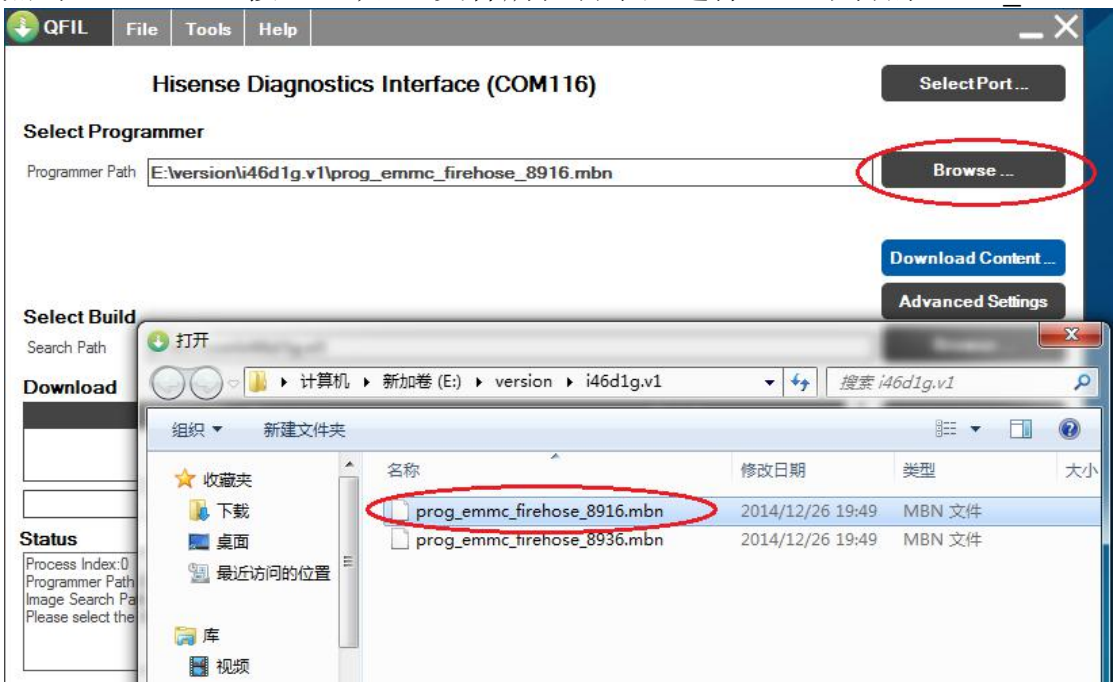


图4-4

3 点击“Load XML”，选择对应的“rawprogram_unsparse.xml”；

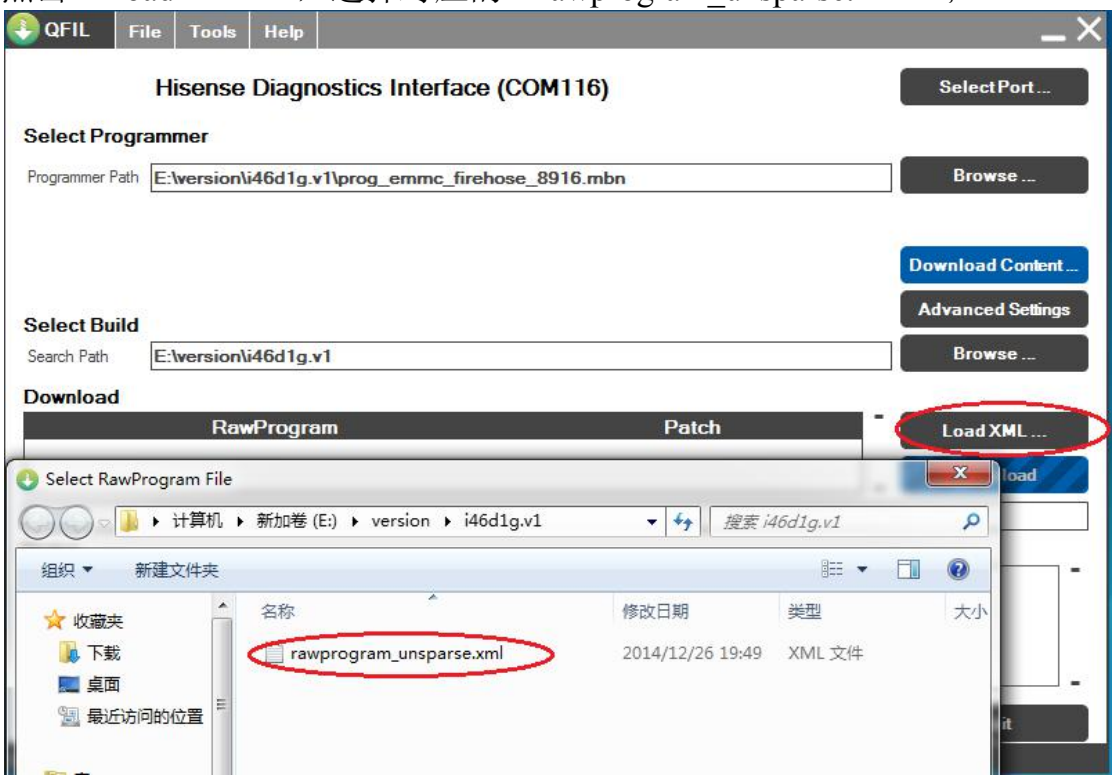


图4-5

4 选择对应的“patch0.xml”文件；

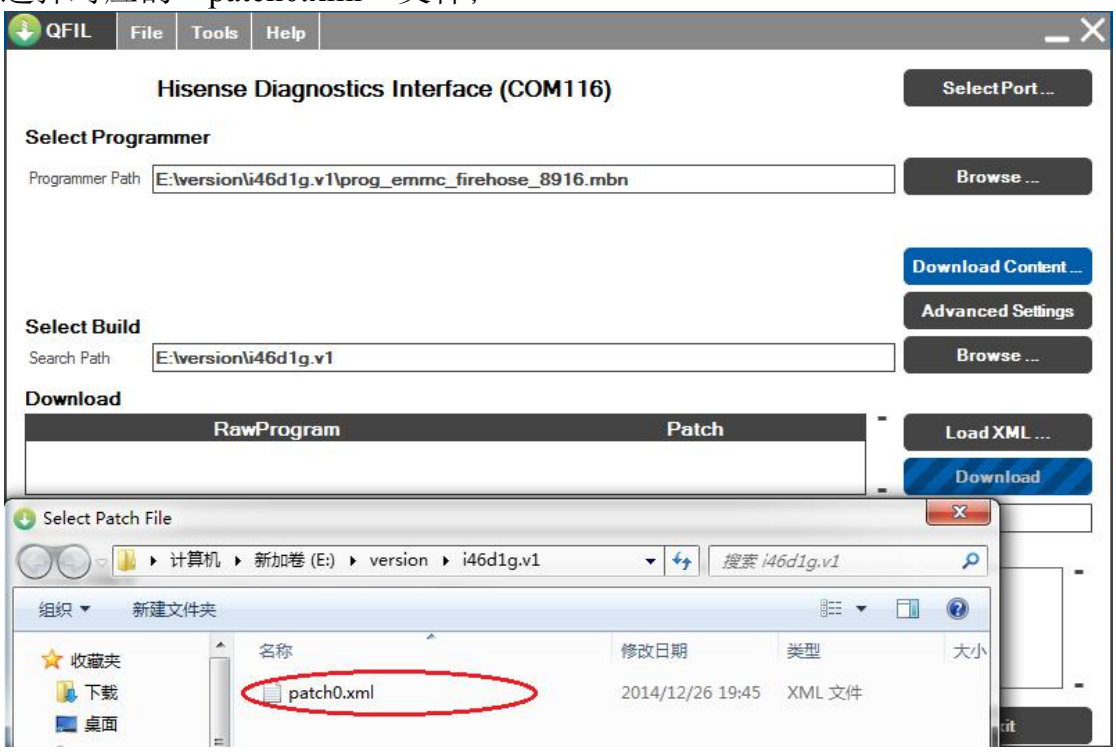


图4-6

5 点击“Download”按钮开始下载。

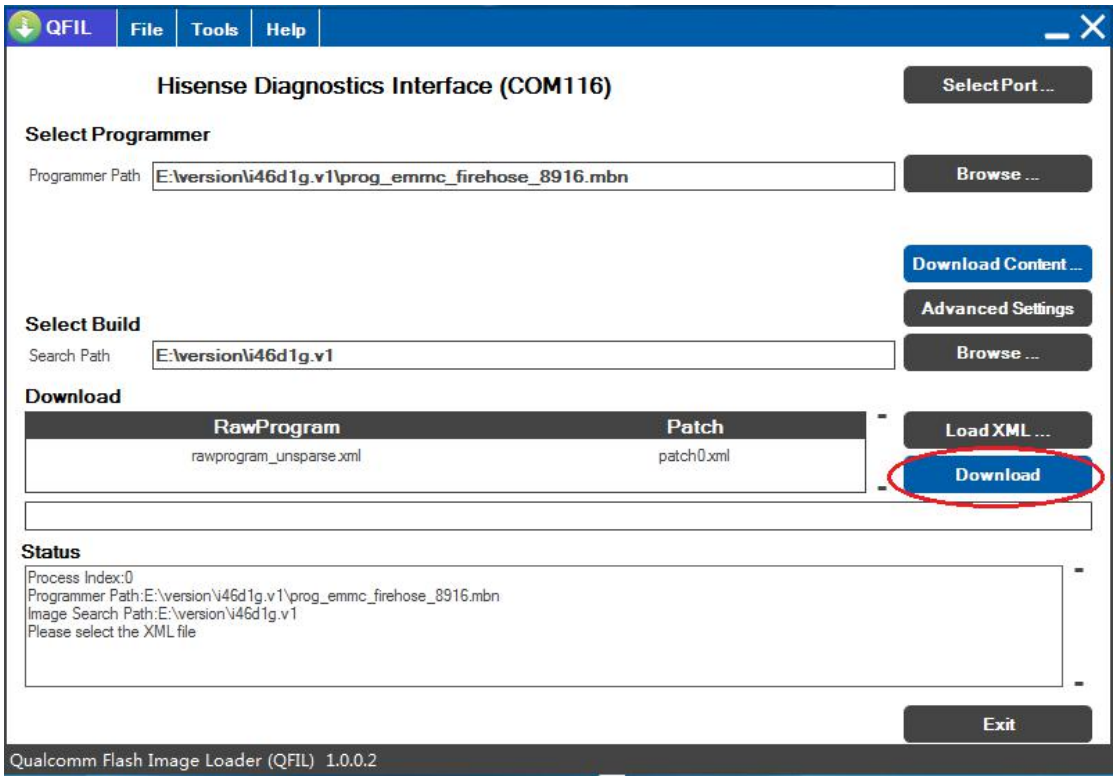


图4-7

4.3 使用TF卡下载Bin文件

1 将需要烧录的系统版本Bin文件拷贝到TF卡根目录，并插入开发板TF卡卡槽；
2 上电，并同时按住上下音量键按钮和电源键按钮进入TF卡下载模式；

4.4 使用Fastboot下载Bin文件

1 使用前请正确安装USB驱动；
2 在关机状态下，按住上音量键然后插入USB线，进入Fastboot下载模式；
3 在PC端执行fastboot命令进行image文件烧录，支持boot.img和system.img的烧录。如图4-8。

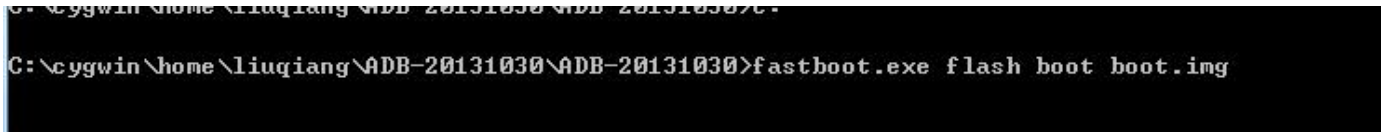


图4-8

5 编译

5.1环境配置

在android根目录下，配置编译环境并选择需要编译的产品，如图5-1，5-2：

```
root@hisense-vpc:~/work1/msm/opsrc/android# . build/envsetup.sh
including device/qcom/common/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including vendor/qcom/proprietary/common/vendorsetup.sh
including sdk/bash_completion/adb.bash
root@hisense-vpc:~/work1/msm/opsrc/android#
```

图5-1

```
root@hisense-vpc:~/work1/msm/opsrc/android# choosecombo
Build type choices are:
     1. release
     2. debug

Which would you like? [1] release


Which product would you like? [full] full


Variant choices are:
     1. user
     2. userdebug
     3. eng
Which would you like? [eng] user

============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.4.4
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=user
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
TARGET_CPU_VARIANT=generic
HOST_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.13.0-32-generic-x86_64-with-Ubuntu-14.04-trusty
HOST_BUILD_TYPE=release
BUILD_ID=KTU84P
OUT_DIR=out
============================================
```

图5-2

5.1 Kernel编译

配置编译环境完毕后，在android根目录执行make kernel,即可完成内核image编译，相应的boot.img在./android/out/target/product/<product name>/目录下保存。如图5-3。

```
root@hisense-vpc:~/work1/msm/opsrc/android# make kernel
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.4.4
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=user
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
TARGET_CPU_VARIANT=generic
HOST_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.13.0-32-generic-x86_64-with-Ubuntu-14.04-trusty
HOST_BUILD_TYPE=release
BUILD_ID=KTU84P
OUT_DIR=out
============================================
```

图5-3

5.2 Lk编译

配置编译环境完毕后，在android根目录执行make aboot,即可完成lk image编译，相应的emmc_appsboot.mbn在./android/out/target/product/<product name>/目录下保存。如图5-4。

```
root@hisense-vpc:~/work1/msm/opsrc/android# make aboot
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.4.4
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=user
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
TARGET_CPU_VARIANT=generic
HOST_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.13.0-32-generic-x86_64-with-Ubuntu-14.04-trusty
HOST_BUILD_TYPE=release
BUILD_ID=KTU84P
OUT_DIR=out
============================================
```

图5-4

6 外围部品调试指南

6.1 Camera调试指南

6.1.1 简介

6.1.2 摄像头传感器驱动

6.1.2.1 YUV和Bayer Sensor参考驱动

本节罗列MSM8916当前主线代码包含的Bayer和YUV sensor参考驱动。$(MM_CAMERA_DIR)默认为android/vendor/qcom/proprietary/mm-camera/。

Bayer参考驱动

用户空间驱动位于

$(MM_CAMERA_DIR)/mm-camera2/media-controller/modules/sensors/sensor_libs/

imx135_lib.c/h
ov2680_lib.c
ov2720_lib.c
ov9724_lib.c
s5k3l1yx_lib.c

YUV参考驱动

用户空间驱动位于

$(MM_CAMERA_DIR)/mm-camera2/media- controller/modules/sensors/sensor_libs/

sp1628_lib.c
SKUAA-Shengtai-hi256_lib.c
ov5645_lib.c
mt9m114_lib.c

内核空间驱动位于

kernel/drivers/media/platform/msm/camera_v2/sensor

sp1628.c
hi256.c
ov5645.c
mt9m114.c

6.1.2.2 需要修改的文件

这个段落列出移植一个新的摄像头传感器驱动需要修改的文件。

Bayer sensor

dtsi文件为kernel/arch/arm/boot/dts/qcom/目录下的<target>_camera*.dtsi。比如msm8916-camera-sensor-mtp.dtsi。客户需使用如下对应的条目:

```
qcom,camera@0 {
cell-index = <0>;
compatible = "qcom,camera";
. . .
}
```

在vendor/qcom/proprietary/common/config/device-vendor.mk文件中，需为新的驱动加入一个条目。

在 $(MM_CAMERA_DIR)/mm-camera2/media-controller/modules/sensors/module/sensor_init.c 文件中的sensor_libs[] 数组中需添加sensor名称。

用户空间驱动是$(MM_CAMERA_DIR)/ mm-camera2/media-controller/modules/sensors/sensor_libs/<snesor>目录下<sensor>_lib.c/h，比如imx135_lib.c/h。

除了驱动文件，位于驱动文件目录下的Android.mk 文件也需修改。

YUV Sensor

dts文件是位于kernel/arch/arm/boot/dts/qcom/目录下的<target>_camera*.dtsi文件，

比如msm8916-camera-sensor-mtp.dtsi，客户需要在该dtsi文件加入类似于如下例子的一个新条目。

```
qcom,camera@78 {
compatible = "ovti,ov5645";
. . .
}
```

用户空间驱动是$(MM_CAMERA_DIR)/mm-camera2/media-controller/modules/sensors/sensor_libs/<snesor>目录下<sensor>_lib.c/h，比如sp1628_lib.c/h。

除了驱动文件，位于驱动文件目录下的Android.mk 文件也需修改。

内核空间驱动是在kernel/drivers/media/platform/msm/camera_v2/sensor/目录下有<sensor>.c文件，该目录下的Makefile文件也要修改。同时，需要在kernel/arch/arm/configs目录下<target>_defconfig文件中添加CONFIG_<sensor> 定义。

配置

在vendor/qcom/proprietary/common/config/device-vendor.mk文件中，需为新的驱动加入一个条目。

6.1.3 源代码解释

6.1.3.1 内核驱动

6.1.3.1.1 GPIO配置

如下所示，客户可以根据自己的硬件设置配置sensor特有的GPIO.关于各个属性的解释，可以参考以下目录下的msm-cci.txt:

kernel/Documentation/devicetree/bindings/media/video/

```
gpios = <&msmgpio 15 0>,
<&msmgpio 90 0>,
<&msmgpio 89 0>;
qcom,gpio-reset = <1>;
qcom,gpio-standby = <2>;
qcom,gpio-req-tbl-num = <0 1 2>;
qcom,gpio-req-tbl-flags = <1 0 0>;
qcom,gpio-req-tbl-label = "CAMIF_MCLK",
"CAM_RESET1",
"CAM_STANDBY";
qcom,gpio-set-tbl-num = <1 1>;
qcom,gpio-set-tbl-flags = <0 2>;
qcom,gpio-set-tbl-delay = <1000 30000>;
```

对于CCI, 在下面MSM8916专有的GPIO用于数据和时钟传输.因为这两个GPIO只用于CCI, 如果客户使用CCI为I2C通信，请保持现有配置。

```
gpios = <&msm_gpio 29 0>,
<&msm_gpio 30 0>;
qcom,gpio-tbl-num = <0 1>;
qcom,gpio-tbl-flags = <1 1>;
qcom,gpio-tbl-label = "CCI_I2C_DATA0",
"CCI_I2C_CLK0";
```

6.1.3.1.2 时钟相关配置

在dts文件中，对于每一个sensor节点，客户可以如下设置时钟源:

```
clocks = <&clock_gcc clk_mclk0_clk_src>,
            <&clock_gcc clk_gcc_camss_mclk0_clk>;
            clock-names = "cam_src_clk", "cam_clk";
```

这两个属性的顺序是相关的。 clock-name属性的第n个值对应clocks属性的第n个值。 因此在上面的dt片段中, cam_src_clk 对应clk_mclk0_clk_src, cam_clk对clk_gcc_camss_mclk0_clk。

以上设置被时钟模块代码引用，客户一般无需修改设置。

6.1.3.1.3 电源设置

PMIC供电

```
cam_vdig-supply = <&pm8916_s4>;
cam_vana-supply = <&pm8916_l17>;
cam_vio-supply = <&pm8916_l6>;
cam_vaf-supply = <&pm8916_l10>;
```

GPIO控制外部LDO供电

```
gpios = <&msm_gpio 27 0>,
<&msm_gpio 28 0>,
<&msm_gpio 33 0>,
<&msm_gpio 114 0>,
<&msm_gpio 110 0>;
qcom,gpio-reset = <1>;
qcom,gpio-standby = <2>;
qcom,gpio-vdig = <3>;
qcom,gpio-vana = <4>;
qcom,gpio-req-tbl-num = <0 1 2 3 4>;
qcom,gpio-req-tbl-flags = <1 0 0 0 0>;
qcom,gpio-req-tbl-label = "CAMIF_MCLK",
                          "CAM_RESET",
                          "CAM_STANDBY",
                          "CAM_VDIG",
                          "CAM_VANA";
```

6.1.3.1.4 I2C slave地址配置

YUV sensor
在dtsi文件中:

```
qcom,camera@78 {
compatible = "ovti,ov5645";
reg = <0x78 0x0>;
qcom,slave-id = <0x78 0x300a 0x5645>;
```

以上设置的I2C slave地址是八位地址，其中高7位为I2C slave地址，最低位是写标记0。

6.1.3.2 用户空间驱动

6.1.3.2.1 Sensor初始化参数

Sensor初始化参数包括支持模式(2D/3D),安装位置 (前置/后置)和安装角度. 如果安装角度设置成360度，内核的dtsi文件中安装角度将用于驱动中。

```
static struct msm_sensor_init_params sensor_init_params = {
            .modes_supported = CAMERA_MODE_2D_B,
            .position = BACK_CAMERA_B,
            .sensor_mount_angle = SENSOR_MOUNTANGLE_360,
};
```

6.1.3.2.2 Sensor输出配置

Sensor输出配置包括输出格式（Bayer或者YUV）。连接模式 (MIPI 或者parallel,MSM8916只支持MIPI)和raw格式类型。

```
static sensor_output_t sensor_output = {
.output_format = SENSOR_BAYER,
.connection_mode = SENSOR_MIPI_CSI,
.raw_output = SENSOR_10_BIT_DIRECT,
};
```

6.1.3.2.3 Bayer slave配置

Sensor slave配置信息必须提供下面信息.

```
static struct msm_camera_sensor_slave_info sensor_slave_info = {
/*该sensor安装在哪个物理接口上*/
.camera_id = CAMERA_0,
/* I2C slave地址*/
.slave_addr = 0x20,
/* sensor地址类型*/
.addr_type = MSM_CAMERA_I2C_WORD_ADDR,
/* sensor id信息*/
.sensor_id_info = {
/* sensor id寄存器地址*/
.sensor_id_reg_addr = 0x0016,
/* sensor id */
.sensor_id = 0x0135,
},
/* 上下电设置*/
.power_setting_array = {
.power_setting = power_setting,
.size = ARRAY_SIZE(power_setting),
},
```

以上设置的I2C slave地址是八位地址，其中高7位为I2C slave地址，最低位是写标记0。

Sensor的上下电时序以数组的格式写于用户空间驱动的msm_sensor_power_setting数据结构中。

```
static struct msm_sensor_power_setting power_setting[] = {
. . .
}
```

上电时序和下电时序都可以分别加在msm_sensor_power_setting_array数组中。如果power_down_setting/size_down数据成员没有添加,下电时序会使用上电时序的反时序。

```
.power_setting_array = {
.power_setting = power_setting,
.size = ARRAY_SIZE(power_setting),
},
```

power_setting 包含GPIO/CLK/VREG 拉高/拉低还有延时的序列。

```
static struct msm_sensor_power_setting power_setting[] = {
{
.seq_type = SENSOR_VREG,
.seq_val = CAM_VDIG,
.config_val = 0,
.delay = 0, //this delay is in ms
},
{
.seq_type = SENSOR_VREG,
.seq_val = CAM_VANA,
.config_val = 0,
.delay = 0, //this delay is in ms
},
. . .
```

该数据结构被内核空间的msm_camera_power_up()调用来配置sensor的上电时序。请参考kernel/include/media/msm_cam_sensor.h的枚举类型。

6.1.3.2.4 输出尺寸表

输出尺寸表定义在如下的sensor_lib_out_info_t数据结构中, 例如:

```
static struct sensor_lib_out_info_t sensor_out_info[] = {
{
/* full size @ 24 fps*/
.x_output = 4208,
.y_output = 3120,
.line_length_pclk = 4572,
.frame_length_lines = 3142,
.vt_pixel_clk = 360000000,
.op_pixel_clk = 360000000,
.binning_factor = 1,
.max_fps = 24.01,
.min_fps = 7.5,
.mode = SENSOR_DEFAULT_MODE,
},
```

6.1.3.2.5 Sensor寄存器地址

必须参考sensor 规格书填写正确的控制曝光和输出大小的寄存器地址。根据规格书中的描述来决定这个值（粗曝光设定的边界值）。

```
static struct msm_sensor_exp_gain_info_t exp_gain_info = {
.coarse_int_time_addr = 0x0202,
.global_gain_addr = 0x0205,
.vert_offset = 4,
};
```

必须根据sensor规格书中寄存器描述填写正确的输出控制寄存器地址

```
static struct msm_sensor_output_reg_addr_t output_reg_addr = {
.x_output = 0x034C,
.y_output = 0x034E,
.line_length_pclk = 0x0342,
.frame_length_lines = 0x0340,
};
```

6.1.3.2.7 寄存器设定

相机sensor 的寄存器通过I2C来配置使sensor 输出数据，本节中还会介绍通过其他特别的操作方式来实现配置sensor。

初始化设定表现为在相机启动时一组一次性写入的寄存器：

```
static struct msm_camera_i2c_reg_setting init_reg_setting[] =
{
. . .
}
```

Sensor工作时更新曝光设定需要操作许多寄存器（曝光时间，每帧行数，增益），他们必须在同一帧完成更新。这些寄存器都有双buffer，并具有按组更新的功能。表现为所有相关寄存器一起完成更新。当设定grouped parameter hold为1时，写入的寄存器数据被暂存的buffer寄存器中。

```
static struct msm_camera_i2c_reg_setting groupon_settings =
{
. . .
}
```

当设定grouped parameter hold为0时， 曝光寄存器的值会被同时更新，参数的变化会在同一帧生效：

```
static struct msm_camera_i2c_reg_setting groupoff_settings =
{
. . .
}
```

Sensor可以有多种操作模式，例如按四分之一大小预览和全尺寸拍照，不同的分辨率可以按如下方式配置：

```
static struct msm_camera_i2c_reg_setting res0 _settings[] =
{
. . .
}
static struct msm_camera_i2c_reg_setting res1_settings[] =
{
. . .
```

}

AEC算法中实际gain 用于曝光计算，实际gain必须转换成寄存器gain去设置sensor。客户需要参考sensor 的datasheet去编写正确的gain转换函数。

xxxx_real_to_register_gain()

xxxx_register_to_real_gain()

下面函数基于上面的转换函数计算曝光时间和gain。

xxxx_calculate_exposure()

下面函数基于曝光时间和gain的计算值填写正确的寄存器设置。

xxxx_fill_exposure_array()

6.2 LCD调试指南

本节将介绍点亮LCD而如何创建/配置各个部分的代码。其中，针对LCD显示相关模块的功能，主要介绍了KERNEL和LK断的点亮LCD的步骤和方法。此外，在本节中还介绍了一些基本的显示相关的知识。

6.2.1 显示源码分布

显示相关的源码分布在开源驱动，应用用户层(HAL)，用户空间处理的库函数，以及farmework层。Android中，源码的位置分配如下：

kernel/drivers/video/msm/mdss - MDSS driver

kernel/arch/arm/boot/dts/qcom - MDSS DTSI and panel DTSI files

hardware/qcom/display - Display HALs

frameworks/base/services/surfaceflinger - SurfaceFlinger server

vendor/qcom/proprietary/mm-core/display - Proprietary postprocessing libraries and daemon

mdss_fb - Top-level IOCTL/native frame buffer interface

mdss_mdp.c - MDP resources (clocks/irq/bus-bw/power)

mdss_mdp_overlay - Overlay/DMA top-level API

mdss_mdp_ctl - Control is the hardware abstraction to club the (LM + DSPP + ping-pong + interface)

mdss_mdp_pipe - SRC pipe-related handling mdss_mdp_intf_**** - MDP panel interface-related handling

mdss_mdp_pp - Postprocessing-related implementation

mdss_mdp_rotator - Rotator APIs (overlay_set/overlay_play interface)

6.2.2 显示调试过程

本节中将详细介绍LCD显示相关的调试流程，并简单介绍在LCD点亮过程中遇到的一些问题的解决和调试方法，主要针对的是LCD MIPI Panel。，其中主要分为两个部分：KERNEL和LK（Little Kernel）。

6.2.2.1 LCD MIPI Panel点亮流程-Kernel

6.2.2.1.1 LCD Panel整体点亮流程

1.首先需要获取开发板和使用的LCD Panel信息，并且对于将要使用的开发板原理图进行评审和阅读，进而对照使用的LCD Panel部品信息，如管脚进行比对，如在此过程中有任何疑问，请及时与硬件工程师进行沟通。

2.验证LK端是否已经启动，并且ADB/fastboot/serial 等接口已经可以正常工作。如果需要，可能还要连接JTAG端口。

3.为选用的LCD Panel准备dtsi描述文件。（可以参考高通的sample panel）

4.根据上面的描述创建完LCD Panel的DTSI文件后，需要将创建的DTSI文件包含进入平台的设备树中。如下：

A.To choose DSI as the primary interface:

```
&mdss_mdp {
        qcom,mdss-pref-prim-intf = "dsi";
};
```

B.To make the panel node qcom dsi-pref-prim-pan:

```
&mdss_dsi0 {
        qcom,dsi-pref-prim-pan = <&dsi_<vendor>_720p_video>;
}
```

5.如果首先调试KERNEL端代码，点亮LCD，需要关闭连续显示功能并关闭LK端的显示，否则KERNEL端不能正常点亮LCD。关闭连续显示和LK端的方法如下：

A.To disable the LK display, change DISPLAY_SPLASH_SCREEN to 0 at bootable/bootloader/lk/target/***/rules.mk.

B.To disable the continuous splash screen in kernel in your panel DTSI file:

```
&dsi_<vendor>_720p_video{
        // qcom,cont-splash-enabled; // disable continuous splash screen
};
```

6.点亮和调试LCD过程中，添加Panel相关的配置文件后，需要确认背光在正确的时间和方式下打开，并且保证在整个的启动过程中，背光一直是打开状态。

7.连接LCD Panel，查验其是否可以正常的启动（点亮）。

6.2.2.1.2 LCD Panel属性介绍

本节中将会针对两种LCD Panel类型的属性进行简要的介绍，这些属性在LCD Panel点亮和调试过程中起着重要的作用，只有正确的配置和使用，LCD Panel才会正常的显示。

1. DSI Video mode panel

如果按照上述步骤操作后，依然不能点亮LCD（LCD Panel无显示），或者LCD Panel可以点亮显示，但是显示内容有一些移位，如上移或下移，此时需要关注LCD Panel的porch值得设定。

The PHY timings can be adjusted accordingly;

qcom,mdss-dsi-h-sync-pulse: Specifies the pulse mode option for the panel.

0 = Don't send hsa/he following vs/ve packet(default)

1 = Send hsa/he following vs/ve packet

qcom,mdss-dsi-hfp-power-mode :Boolean to determine DSI lane state during horizontal front porch (HFP) blanking period.

qcom,mdss-dsi-hbp-power-mode: Boolean to determine DSI lane state during horizontal back porch (HBP) blanking period.

qcom,mdss-dsi-hsa-power-mode: Boolean to determine DSI lane state during horizontal sync active (HSA) mode.

2. DSI Command mode panel

a)If you see "garbage" on the display, check your TE pulse using a scope and verify that it is performing at 60 fps.

b) Verify that you are using the right GPIO for TE signal. Each MSM? has a specific GPIO for TE.

c)To rule out that the TE is the issue, try using software Vsync by removing these lines from the panel DTSI:

qcom,mdss-dsi-te-pin-select = <1>;

qcom,mdss-dsi-te-check-enable;

qcom,mdss-dsi-te-using-te-pin;

6.2.2.1.3 LCD Panel点亮注意事项

大多数的LCD Panels都具有自己的reset时序，因此，在DTSI属性配置中需要根据实际的LCD Panel IC特性设定其Reset时序，设定的方法如下：

qcom,mdss-dsi-reset-sequence: An array that lists the sequence of reset gpio values and sleeps Each command will have the format defined as below:
--> Reset GPIO value
--> Sleep value (in ms)

例如，LCD Panel的Reset 需要首先拉高20ms，然后拉低200ms，再接着拉高20ms，之后一直保持在High,Reset时序的设置如下：
qcom,mdss-dsi-reset-sequence = <1 20>, <0 200>, <1 20>;
当前多数LCD Panels需要在LP11状态下做Reset动作，即在做Reset之前需要首先时能DSI CLOCK和data lanes。DTSI属性设置中如下可以达到上述要求：

qcom,mdss-dsi-lp11-init:Boolean used to enable the DSI clocks and data lanes (low power 11) before issuing hardware reset line.
Some panels have requirements to first send one frame of data followed by the ON commands to avoid showing a garbage screen. This DTSI property helps to achieve it:
qcom,mdss-dsi-on-command-state :
This can be set to "dsi_lp_mode" or "dsi_hs_mode" depending on whether the ON commands need to be sent before OR after HS data transmission begins.
Some panels have requirements to send the OFF commands before turning OFF high-speed data transmission to avoid any fade-out effect. The following property helps achieve this:
qcom,mdss-dsi-off-command-state :
This can be set to "dsi_lp_mode" or "dsi_hs_mode" depending on whether the OFF commands need to be sent after OR before HS data transmission stops.

某些平台下，需要设置MIPI CLK始终保持的高速的模式下，设置方法如下：

```
int mdss_dsi_on(struct mdss_panel_data *pdata)
{
        ************************************************
        if (mipi->force_clk_lane_hs) {
                u32 tmp;
                tmp = MIPI_INP((ctrl_pdata->ctrl_base) + 0xac);
                tmp |= (1<<28);
                MIPI_OUTP((ctrl_pdata->ctrl_base) + 0xac, tmp);
                wmb();
        }
}
```

在以上代码片断中设置mipi->force_clk_lane_hs = 1。

6.2.2.2 LCD MIPI Panel点亮流程-Lk

根据高通平台的显示流程，当前建议的是首先在KERNEL端调试点亮LCD Panel，然后再调试点亮LK端。

1.在调试LK端的LCD Panel时，首先需要确认的是主开发板上已经连接有串口或JTAG调试接口，以便于在LK端点亮LCD时调试可能出现的问题，并一定保证背光是常开状态。

2.Create the panel header file using the GCDB tool or by referring to one of the existing header files present in:
bootable/bootloader/lk/dev/gcdb/display/include/

3.LCD Panel header file创建后，将其添加到上述的目录中（in step 2）。

4.Assuming that the panel is already up in kernel, you can either use the same DSI PHY timings as kernel. If kernel is not up, to calculate the timings and add them to the .xml file or directly in the header file if you are not using GCDB.

5.After making the header file, include it in
bootable/bootloader/lk/target/msm8916/oem_panel.c
add <Vendor>_<Resolution>_VIDEO_PANEL case on init_panel_data function

6.连接LCD Panel到开发板上，正常启动开发板，查看其是否可以正常启动。
a).如果不能正常启动，那么通过串口或者JTAG口打印相关LOG信息并查看错误信息，解决错误信息，重新验证。
b).如果解决了log中的错误信息，LCD Panel还是不能正常启动，那就需要dump出来LK中的DSI寄存器值，并与KERNEL中的进行对比（此时KERNEL中已经正常点亮并显示LCD Panel）。
在LK中dump LCD DSI 寄存器的值需要适用JTAG。下面描述中红色字体表示的DSI寄存器的基地址。

```
mdss_dsi0: qcom,mdss_dsi@fd922800 {
        compatible = "qcom,mdss-dsi-ctrl";
        label = "MDSS DSI CTRL->0";
        cell-index = <0>;
        reg = <0xfd922800 0x1f8>,
        mdss_dsi1: qcom,mdss_dsi@fd922e00 {
        compatible = "qcom,mdss-dsi-ctrl";
        label = "MDSS DSI CTRL->1";
        cell-index = <1>;
        reg = <0xfd922e00 0x1f8>,
```

6.2.3 更换开机Logo

接下来介绍一下怎样更换开机Logo（开机显示的第一张图片的方法）。

当前开机Logo的添加和显示都是在LK端完成实现的，因此开机Logo的更换和显示需要在LK端完成。

首先需要指出的是Logo图片的格式，以720P显示为例：

图片大小： 720X1280
图片格式： png
图片位深： 24

制作好上述图片后，更换开机Logo图片的话，直接将图片命名为logo，然后将logo图片拷贝到一下目录：

bootable/bootloader/lk/target/PROJECT_NAME/logo

更换后直接编译LK，下载到手机中即可。

6.3 TP调试指南

6.3.1 TP 源码分布

TP源码位置主要在kernel端，源码的位置分配如下：

```
kernel/drivers/input/touchscreen/ts_func_ts.c -  工检相关接口
kernel/drivers/input/input.c                    ---输入子系统相关代码
kernel/drivers/input/touchscreen/focaltech/   ---敦泰代码
kernel/drivers/input/touchscreen/goodix/      ---汇顶代码
```

6.3.2 TP 调试流程

6.3.2.1 TP 使能流程介绍

1.首先需要获取开发板和使用的TP Panel信息，并且对于将要使用的开发板原理图进行评审和阅读，进而对照使用的TP Panel部品信息，如管脚进行比对，如在此过程中有任何疑问，请及时与硬件工程师进行沟通。

2.为选用的TP Panel准备dtsi描述文件。（可以参考高通的sample panel）dtsi文件一般路径存放在kernel/arch/arm/boot/dts/hisense/<xxx>/xxx.dtsi中。

3.根据硬件描述配置项目dtsi，例如：

```
i2c@78b9000 { /*BLSP1 QUP5*/
focaltech@38 {
            compatible = “focaltech,5x06”;
            reg = <0x38>;
            interrupt-parent = <&msm_gpio>;
            interrupts = <13 0x2>;
            vdd-supply = <&pm8916_l17>;
            vcc_i2c-supply = <&pm8916_l6>;
            focaltech,reset-gpio = <&msm_gpio 12 0x0>;
            focaltech,irq-gpio = <&msm_gpio 13 0x2>;
            focaltech,display-coords = <0 0 720 1280>;
            focaltech,panel-coords = <0 0 720 1280>;
            focaltech,button-map = <139 172 158>;
}
```

主要根据原理图设置vcc，vdd，reset int管脚，及分辨率，按键坐标。

4.在probe函数中对相关硬件配置进行解析配置，上电，中断申请，测试i2c是否通信。

解析dts：

```
err = ft5x06_parse_dt(&client->dev,pdata);
```

上电：

```
err = ft5x06_power_on(data,true);
```

中断申请：

```
err = requested_threaded_irq(client->irq,NULL,
                        ft5x06_ts_interrupt,
                        pdata->irqflags | IRQF_ONESHOT,
                        client->dev.driver->name,data);
```

检测I2C通信：

```
ft5x06_i2c_read(client,&reg_addr,1,&reg_value,1);
```

5.连接TP，查验其是否可以正常的启动（点亮）。

6.3.2.2 TP 工检功能介绍

本小节中将会针对TP工检相关接口进行简要的介绍。

1.项目会提供以下功检接口，以便工艺进行功能检查。对于有些芯片不支持的功能节点，一律显示“NOT SUPPORTED”（注意：大写）。

（1）/sys/ctp/ctp_test /rawdatashow

只读。显示有效的rawdata数据集合。

（2）/sys/ctp/ctp_test/rawdatainfo

只读。显示rawdata的有效通道数和上下限信息。格式为“RX:15 TX:25 HIGH:10000 LOW:6500”。一般情况下，带按键的触屏，有一个通道是用于按键区的，我们的功检中，触屏按键是与其他物理按键（power/vol+/vol-）一同检查的，不是通过rawdata检查。因此，有效通道数应当是总通道数减去按键通道。

（3）/sys/ctp/ctp_test/resettest

只读。显示reset信号线检测结果，成功或者失败。0：失败；1：成功。

（4）/sys/ctp/ctp_test/fwversion

只读。显示当前IC中固件版本号，如0X0B。

注：版本号为4字节数据，如含字母，必须为大写。

（5）/sys/ctp/ctp_test/fwhostversion

只读。显示版本集成的固件版本号。

注：版本号为4字节数据，如含字母，必须为大写。

（6）/sys/ctp/ctp_test/fwmoduleid

只读。显示模组ID。

（7）/sys/ctp/ctp_test/caltest

只读。校准标志，显示该TP是否进行过校准。0：未校准；1：已校准。

2、工检节点建立在kernel/drivers/input/ts_func_test.c中，在IC相关源代码中实现相关接口，并在probe函数中调用如下函数进行注册。

```
factory_ts_func_test_register(data);
```

6.3.2.3 TP 固件升级介绍

本节对TP panel固件升级功能做简单介绍。

1. TP固件升级共有两种升级方法。自动升级及T卡升级。

1）自动升级：在开机流程中加入触屏固件升级的检查。开机时，先判断触屏固件是否需要升级，如果需要升级，显示提示界面并开始升级，升级完成后重启或继续开机流程；如果不需要升级，则继续开机流程。上层需要实现功能调用及显示，kernel中需要添加两个功能节点，如下：

（1）/sys/ctp/ctp_update/fwstate

只读。0：TP固件不需要升级；1：TP固件需要升级。

（2）/sys/ctp/ctp_update/fwupdate

读写。

读取该节点，0：升级失败；1：升级成功；2：升级中；向该节点写入“UPDATE”启动触屏固件升级功能。

注：节点权限设置为644，在init.rc中设置owner为system，以免影响CTS测试。

2）T卡升级：为了能够在产线只升级触屏固件，而不必重新升级大软件版本，需要实现T卡升级触屏固件的功能。方法是，将固件放置于T卡内，进入功检，功检程序将固件存放路径写入kernel提供的升级节点触发固件升级。对于kernel，具体要求如下：

/sys/ctp/ctp_update/fwbinupdate

读写。

读取该节点，0：升级失败；1：升级成功；2：升级中；向该节点写入固件存放路径，启动触屏固件升级。如：

echo /sdcard/0x53_0x67_20131016.bin > /sys/ctp/ctp_update/fwbinupdate。

注：节点权限设置为644，在init.rc中设置owner为system，以免影响CTS测试。

该节点在建立在kernel/drivers/input/ts_func_test.c，在IC相关源代码中实现相关接口：

```
Data->ts_test_dev.check_fw_update_need = factory_check_fw_update_need;
Data->ts_test_dev.proc_fw_update = factory_proc_fw_update;
```

并在probe函数中调用如下接口进行注册。

```
factory_ts_func_test_register(data);
```

6.4 UART

TBD

6.5 USB

TBD

6.6 GPIO

TBD

7 推荐部品列表

7.1 LCD/CTP

LCD/CTP推荐部品列表请参考表7-1。

	厂商	IC 型号
LCD		
	HIMAX	HX8394A, HX8394D
	ORISE	OTM1283A, OTM1287A
CTP		
	GOODIX	GT915L, GT9157
	FOCALTECH	FT5336, FT5446

表7-1

7.2 Camera

Camera推荐部品列表请参考表7-2。

厂商	IC 型号
Omnivision	
	OV8865（8M）
	OV5648（5M）
	OV2680（2M）
Sony	
	IMX219（13M）
	IMX214（8M）
Samsung	
	S5K3M2（13M）
	S5K5E2（5M）

表7-2

7.3 传感器

传感器推荐部品列表请参考表7-3。

	厂商	IC 型号
加速度传感器		
	KIONIX	KXTIK1004
	BOSCH	BMA222/223
地磁传感器		
	AKM	AK09911
距离光传感器		
	AVAGO	APDS9930
	CAPELLA	CM36686

表7-3