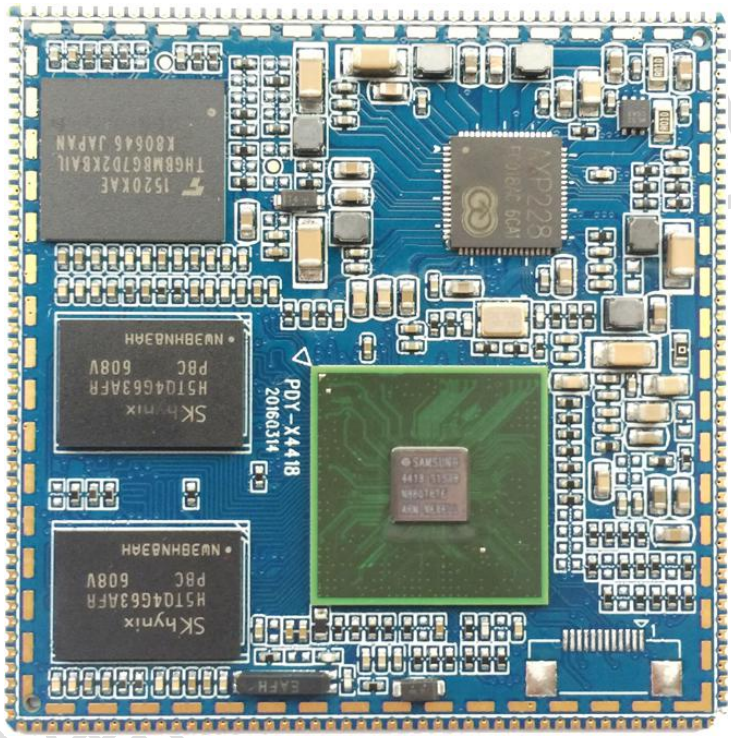
G4418 Linux QT 平台用户手册



深圳葡萄雨技术有限公司

www.graperain.cn

版权声明

敬告：在售开发平台的手册会经常更新，请在http://www.graperain.cn/网站下载最新手册或与我司销售联系取得，不再另行通知。

版本说明

版本号	日期	作者	描述
Rev.01	2016-4-7	David Huang	修订版本

技术支持

如果您对文档有所疑问，您可以拨打技术支持电话或 E-mail 联系。

网　　址：**http://www.graperain.cn/**

联系电话：0755-23025312

E-mail：info@graperain.com

销售与服务网络

公司：深圳市葡萄雨技术有限公司

地址：深圳市宝安区西乡街道银田路 4 号

邮编：518101

电话：0755-23025312

网址：**http://www.graperain.cn/**

目录

版权声明……错误！未定义书签。
第 1 章 Linux QT 平台开发环境搭建……3
1.1 使用 U 盘安装 ubuntu……3
1.2 安装相关依赖包……5
1.3 安装交叉编译工具……5
1.4 指定 GCC 交叉编译器……6
第 2 章 编译 Linux QT 源码包……8
2.1 安装 Linux QT 源码包……8
2.2 编译脚本分析……9
2.3 编译源码……13
2.3.1 查看编译帮助……13
2.3.2 编译 uboot……13
2.3.3 编译 kernel……14
2.3.4 编译 Linux QT 文件系统……14
第 3 章 制作启动卡……15
3.1 Ubuntu 下制作启动卡……15
3.2 Windows 下制作启动卡……18
第 4 章 烧写 Linux QT 映像文件……24
4.1 空板升级……24
4.2 正常升级……25
4.2.1 使用 TF 卡脱机升级……25
4.2.2 Windows 下使用 fastboot 升级……26

4.2.3 Ubuntu 下使用 fastboot 升级……28
4.2.4 Uboot 环境变量设置……34
第 5 章 Linux QT 测试程序……36
5.1 测试 LED……错误！未定义书签。
5.2 测试蜂鸣器……错误！未定义书签。
5.3 调节背光……错误！未定义书签。
5.4 按键测试……错误！未定义书签。
5.5 测试 ADC 电压……错误！未定义书签。
5.6 音频测试……错误！未定义书签。
5.7 测试触摸屏……错误！未定义书签。
5.8 测试串口……错误！未定义书签。
5.9 摄像头测试……错误！未定义书签。
5.10 测试网络……错误！未定义书签。
5.11 TF 卡测试……错误！未定义书签。
5.12 测试 U 盘……错误！未定义书签。
5.13 测试休眠唤醒……错误！未定义书签。
5.14 测试重启……错误！未定义书签。
5.15 测试关机……错误！未定义书签。
第 6 章 使用 ramdisk 文件系统……37
第 7 章 产品线介绍……38
7.1 核心板系列……38
7.2 开发板系列……38
7.3 卡片电脑系列……38

第 1 章 Linux QT 平台开发环境搭建

对于 Linux QT 平台源码的编译，不像 android 那样，对 PC 机性能要求很高，因此用户可以在虚拟机安装 Ubuntu 操作系统，搭建环境来开发 linux。

对于在 windows 操作系统中如何安装虚拟机及在虚拟机中安装 Ubuntu 操作系统，用户可以网上查看相关资料，或者查阅我们的文档《使用虚拟机安装 ubuntu 系统》。

而为了编译效率，充分发挥 PC 的性能，推荐用户直接安装 Ubuntu 操作系统；下面我们讲述的就是此种方式。在开发中，我们使用的 Ubuntu14.04 64 位系统，建议用户与我们所用的版本保持一致。

1.1 使用 U 盘安装 ubuntu

使用 U 盘安装 Ubuntu 系统，简单便捷，推荐使用此方法进行安装。

相关工具：

容量大小 2GB 以上的 U 盘一个

Linuxlive usb creater 软件，下载地址：http://www.linuxliveusb.com/

ubuntu14.04 系统，下载地址：http://www.ubuntu.com/download/desktop/

安装方法：

第一步：下载好 ubuntu 的 ISO 文件；下载linuxlive usb creater 软件并安装。

第二步：插入 u 盘，并打开 usb creater 软件，根据软件提示设置，在步骤 1 中选择安装盘，找到识别出的 U 盘；在步骤 2 中找到下载的 ubuntu 映像文件；步骤 3 默认，步骤 4 中 选中隐藏优盘上创建的文件，使用 FAT32 格式化 U 盘；最后在步骤 5 中点击闪电图标开始安装，直到提示优盘已安装完成为止。（如下图）

第三步：重启电脑，开机时，进入 BIOS 设置菜单，选择 U 盘启动。
一般台式机是按 DEL 键，笔记本有些是按 F2，有些按 F10 进入。设置完成后保存退出。（以下过程供参考）
第四步：再次重启系统，这时已经可以看到 ubuntu 的安装界面，选择中文，继续；
第五步：选择 install，继续；
第六步：选择中文，点击继续；
第七步：配置网络，可以安装时升级，也可以不升级，待安装完系统后再手动升级；
第八步： 选something else继续在这里分出了两个区给 ubuntu，一个 / 和 一个 /home ， 分区可以新建，可以对它格式化，具体根据需要选择；
第九步：设置区域，选择上海；
第十步：选择键盘布局，选中国；

第十一步 :选择用户名和密码 ,到此配置完毕 ,点击继续直接安装 ,一段时间过后 ,安装完毕，

重启之后，进入到 ubuntu 系统。

1.2 安装相关依赖包

说明 :本文档所有开发全部基于 **ubuntu14.04 64** 位系统 ,后续不再声明。
依赖的软件包及64位系统补丁包：

git ,git-core ,gnupg ,flex ,bison ,gperf ,libsdl-dev ,libesd0-dev ,libwxgtk2.6-dev ,

libwxgtk2.8-dev ,build-essential ,zip ,curl ,libncurses5-dev ,zlib1g-dev ,genromfs ,

lsb-core , libc6-dev-i386 , g++-multilib , lib32z1-dev , lib32ncurses5-dev ,

lib32stdc++-4.9-dev , lib32z1 , u-boot-tools, android-tools-fastboot

使用如下命令安装所需的软件包：

sudo apt-get update
sudo apt-get upgrade

sudo apt-get install git git-core gnupg flex bison gperf libsdl-dev libesd0-dev

libwxgtk2.6-dev libwxgtk2.8-dev build-essential zip curl libncurses5-dev zlib1g-dev

genromfs lsb-core libc6-dev-i386 g++-multilib lib32z1-dev lib32ncurses5-dev

u-boot-tools android-tools-fastboot lib32stdc++-4.9-dev lib32z1

以上软件包，建议逐个去安装，这样可以发现哪个安装失败。

1.3 安装交叉编译工具

交叉编译工具链已经集成到源码包中，无需再手动安装。交叉编译工具链路径：（android源码中）

prebuilts/gcc/linux-x86/arm/arm-eabi-4.7

1.4 指定 GCC 交叉编译器

在 ubuntu 系统上安装最新的 GCC 交叉编译器时，版本已经超过 4.4 了，使用如下指令可查询 GCC 的版本：

gcc –version

可能出现的如下信息：

david@ubuntu-server:~$ gcc --version

gcc (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.8.3

Copyright © 2011 Free Software Foundation, Inc.

上面显示 4.8版本，默认我们给出的包，在 4.8上编译会提示一些错误，都是新的GCC引出的错误，网上都有解决办法，如果不想修改这些错误，可将 GCC 版本降至 4.4 即可

解决办法：

安装4.4版本

sudo apt-get install gcc-4.4 g++-4.4 g++-4.4-multilib

装完后，开始降级 gcc

cd /usr/bin

sudo mv gcc gcc.bk

sudo ln -s gcc-4.4 gcc

sudo mv g++ g++.bk

sudo ln -s g++-4.4 g++

再次查看版本，降级完成：

david@david-work:~$ gcc -version

gcc: unrecognized option '-version'

gcc: no input files

david@david-work:~$ gcc --version

gcc (Ubuntu/Linaro 4.4.7-8ubuntu1) 4.4.7

Copyright (C) 2010 Free Software Foundation, Inc.

第 2 章 编译 Linux QT 源码包

g4418 开发板及 gbox4418 卡片电脑标配 emmc 存储芯片。

说明：编译映像时一定要使用普通权限编译。编译完成后，最终只需要三个映像文件 **ubootpak.bin**，**boot.img**，**qt-rootfs.img**。

ubootpak.bin: bootloader 用于引导内核 它包含了 2ndboot.bin ,nsih 以及 u-boot.bin

三个文件，将它们打包成一个文件，方便调试及更新映像。

boot.img：它包含了内核 uImage 以及 ramdisk.img 两个文件。

- qt-rootfs.img：linux qt 文件系统映像

说明：更新 **boot.img**，等同于同时更新了 **uImage** 和 **ramdisk**。

2.1 安装 Linux QT 源码包

从网盘中拷贝 linux 源码包，放在自己的用户名目录，网盘中存放着linux系统的源码包，其名称为 g4418_linux-20160420.tar.bz2。注意最好不要放在文件系统的根目录，这样会出现管理权限问题。

示例方法：在用户权限下执行如下命令：

```
cp  yourcdromdir/source/ g4418_linux-20160420.tar.bz2   ~/

cd

tar  xvf  g4418_linux-20160420.tar.bz2
```

解压完成后，在目录中有4个文件夹：
uboot： 存放的是uboot源码；
kernel：存放的是内核源码；
buildroot：存放的是linux文件系统源码；
linux：存放uboot，内核源码，一些依赖库、功能库和测试程序，等；

至此，Linux QT 源码包安装完成。

说明：

1、源码包名称可能会因发布日期等有所不同，具体以光盘中实际名称为准。本源码包 同

时支持 g4418开发板和 gbox4418卡片电脑。

2、linux 系统的 uboot，内核和 android 的可以直接兼容，linux 系统中使用电容触摸屏时，只需改动 uboot 中启动环境变量 bootargs 即可，详细请参考 uboot 环境变量设置章节。

2.2 编译脚本分析

说明：各种版本的源码编译脚本大同小异，原理完全相同，具体脚本以相关源码包中的为准，这里仅用来分析其实现机制。

编译脚本文件 mk内容及注释如下：

```
#!/bin/bash

#
# JAVA PATH
#
export PATH=/usr/lib/jvm/java-6-oracle/bin:$PATH

#
# Some Directories
#
BS_DIR_TOP=$(cd `dirname $0`; pwd)
BS_DIR_RELEASE=${BS_DIR_TOP}/out/release
BS_DIR_TARGET=${BS_DIR_TOP}/out/target/product/g4418/
BS_DIR_UBOOT=${BS_DIR_TOP}/linux/bootloader/u-boot-2014.07
BS_DIR_KERNEL=${BS_DIR_TOP}/linux/kernel/kernel-3.4.39
BS_DIR_BUILDROOT=${BS_DIR_TOP}/buildroot

#
# Cross Toolchain Path
# 声明 uboot 和内核的交叉编译工具
BS_CROSS_TOOLCHAIN_BOOTLOADER=${BS_DIR_TOP}/prebuilts/gcc/linux-x86/arm/arm-eabi-4.7/bin/arm-eabi-
BS_CROSS_TOOLCHAIN_KERNEL=${BS_DIR_TOP}/prebuilts/gcc/linux-x86/arm/arm-eabi-4.7/bin/arm-eabi-

#
# Target Config
#指定 uboot，内核以及文件系统的配置文件
BS_CONFIG_BOOTLOADER_UBOOT=g4418_config
BS_CONFIG_KERNEL=g4418_defconfig
BS_CONFIG_FILESYSTEM=PRODUCT-g4418-userdebug
BS_CONFIT_BUILDROOT=g4418_defconfig
```

```
#在编译前，设置一些编译环境，保证可靠地编译
setup_environment()
{
    LANG=C
    PATH=${BS_DIR_TOP}/out/host/linux-x86/bin:$PATH;
    cd ${BS_DIR_TOP};
    mkdir -p ${BS_DIR_RELEASE} || return 1
}

#编译 uboot，编译完后，自动将 ubootpak.bin 拷贝到 out/release 目录
build_bootloader_uboot()
{
    # Compiler uboot
    cd ${BS_DIR_UBOOT} || return 1
    make distclean CROSS_COMPILE=${BS_CROSS_TOOLCHAIN_BOOTLOADER} ||
return 1
    make ${BS_CONFIG_BOOTLOADER_UBOOT}
CROSS_COMPILE=${BS_CROSS_TOOLCHAIN_BOOTLOADER} || return 1
    make -j${threads} CROSS_COMPILE=${BS_CROSS_TOOLCHAIN_BOOTLOADER}
|| return 1

    # Copy bootloader to release directory
    cp -v ${BS_DIR_UBOOT}/ubootpak.bin ${BS_DIR_RELEASE}
    cp -v ${BS_DIR_UBOOT}/readme.txt ${BS_DIR_RELEASE}
    cp -v ${BS_DIR_UBOOT}/env.txt ${BS_DIR_RELEASE}
    cp -v ${BS_DIR_UBOOT}/s5p4418-sdmmc.sh ${BS_DIR_RELEASE}

    echo "^_^ uboot path: ${BS_DIR_RELEASE}/ubootpak.bin"
    return 0
}

#编译内核，编译完成后，会自动将内核映像 uImage 拷贝到 out/release 目录
#将内核映像 uImae 和 ramdisk 文件系统打包成 boot.img，并复制到 out/release 目录
build_kernel()
{
    # Compiler kernel
    cd ${BS_DIR_KERNEL} || return 1
    make ${BS_CONFIG_KERNEL} ARCH=arm
CROSS_COMPILE=${BS_CROSS_TOOLCHAIN_KERNEL} || return 1
    make -j${threads} ARCH=arm
CROSS_COMPILE=${BS_CROSS_TOOLCHAIN_KERNEL} || return 1
    make -j${threads} ARCH=arm
CROSS_COMPILE=${BS_CROSS_TOOLCHAIN_KERNEL} uImage || return 1

    # Copy uImage to release directory
    cp -v ${BS_DIR_KERNEL}/arch/arm/boot/uImage ${BS_DIR_RELEASE}

    echo "^_^ kernel path: ${BS_DIR_RELEASE}/uImage"

    # generate boot.img
    cd ${BS_DIR_TOP} || return 1
    echo 'boot.img ->' ${BS_DIR_RELEASE}
    # Make boot.img with ext4 format, 64MB
    cp -v ${BS_DIR_RELEASE}/uImage ${BS_DIR_TARGET}/boot
    mkuserimg.sh -s ${BS_DIR_TARGET}/boot ${BS_DIR_TARGET}/boot.img ext4
boot 67108864

    cp -av ${BS_DIR_TARGET}/boot.img ${BS_DIR_RELEASE} || return 1;
```

```
    return 0
}

#   编译 android 文件系统
build_system()
{
    cd ${BS_DIR_TOP} || return 1
    source build/envsetup.sh || return 1
    make -j${threads} ${BS_CONFIG_FILESYSTEM} || return 1

    # Make boot.img
    # Create boot directory
    mkdir -p ${BS_DIR_TARGET}/boot || return 1

    # Copy some images to boot directory
    if [ -f ${BS_DIR_RELEASE}/uImage ]; then
        cp -v ${BS_DIR_RELEASE}/uImage ${BS_DIR_TARGET}/boot
    fi
    if [ -f ${BS_DIR_TARGET}/ramdisk.img ]; then
        cp -v ${BS_DIR_TARGET}/ramdisk.img ${BS_DIR_TARGET}/boot/root.img.gz
    fi
    if [ -f ${BS_DIR_TARGET}/ramdisk-recovery.img ]; then
        cp -v ${BS_DIR_TARGET}/ramdisk-recovery.img ${BS_DIR_TARGET}/boot
    fi

    # Make boot.img with ext4 format, 64MB
    mkuserimg.sh -s ${BS_DIR_TARGET}/boot ${BS_DIR_TARGET}/boot.img ext4
boot 67108864

    # Copy to release directory
    cp -av ${BS_DIR_TARGET}/boot.img ${BS_DIR_RELEASE} || return 1;
    cp -av ${BS_DIR_TARGET}/system.img ${BS_DIR_RELEASE} || return 1;
    cp -av ${BS_DIR_TARGET}/cache.img ${BS_DIR_RELEASE} || return 1;
    cp -av ${BS_DIR_TARGET}/recovery.img ${BS_DIR_RELEASE} || return 1;
    cp -av ${BS_DIR_TARGET}/userdata.img ${BS_DIR_RELEASE} || return 1;

    return 0
}

#   编译 linux + qt 文件系统
build_buildroot()
{
    # Compiler buildroot
    cd ${BS_DIR_BUILDROOT} || return 1
    make ${BS_CONFIT_BUILDROOT} || return 1
    make || return 1

    # Copy image to release directory
    cp -v ${BS_DIR_BUILDROOT}/output/images/rootfs.ext4
${BS_DIR_RELEASE}/qt-rootfs.img
    cp -v ${BS_DIR_BUILDROOT}/qt-documents.txt ${BS_DIR_RELEASE}
}

threads=1
uboot=no
kernel=no
```

```
system=no
buildroot=no

if [ -z $1 ]; then
	uboot=yes
	kernel=yes
	system=yes
	buildroot=yes
fi

while [ "$1" ]; do
	case "$1" in
	-j=*)
		x=$1
		threads=${x#-j=}
		;;
	-u|--uboot)
		uboot=yes
		;;
	-k|--kernel)
		kernel=yes
		;;
	-s|--system)
		system=yes
		;;
	-b|--buildroot)
		buildroot=yes
		;;
	-a|--all)
		uboot=yes
		kernel=yes
		system=yes
		buildroot=yes
		;;
	-h|--help)
		cat >&2 <<EOF
Usage: build.sh [OPTION]
Build script for compile the source of telechips project.

  -j=n                    using n threads when building source project (example:
-j=16)
  -u, --uboot             build bootloader uboot from source
  -k, --kernel            build kernel from source
  -s, --system            build android file system from source
  -b, --buildroot         build buildroot file system for QT platform
  -a, --all               build all, include anything
  -h, --help              display this help and exit
EOF
		exit 0
		;;
	*)
		echo "build.sh: Unrecognised option $1" >&2
		exit 1
		;;
	esac
	shift
done
```

```
setup_environment || exit 1

if [ "${uboot}" = yes ]; then
    build_bootloader_uboot || exit 1
fi

if [ "${kernel}" = yes ]; then
    build_kernel || exit 1
fi

if [ "${system}" = yes ]; then
    build_system || exit 1
fi

if [ "${buildroot}" = yes ]; then
    build_buildroot || exit 1
fi

exit 0
```

2.3 编译源码

2.3.1 查看编译帮助

执行如下指令可查询 mk 脚本使用方法：

./mk –h

2.3.2 编译 uboot

在 linux 源码目录下执行如下命令编译 uboot 编译完成后映像文件 ubootpak.bin 会释 放到 out/release 目录：

```
./mk  -u
```

说明：默认烧写 **uboot** 时，需要烧写 **2ndboot.bin, nsih** 以及 **u-boot.bin** 三个文件，这里 我们将三个文件打包成一个文件了 并命名为 **ubootpak.bin**。因此，在平时调试时，我们不再需要关注 **2ndboot.bin**，**nsih** 以及 **u-boot.bin** 这三个文件。

注意:

g4418 开发板和 gbox4418 卡片电脑的 uboot 会有区别，默认 g4418 开发板使用的是 16 BIT 的 DDR3，而 gbox4418 使用的是 8 BIT 的 DDR3，因此 DDR 配置会有所差别。

在uboot 源码包的根目录下，存在如下几个文件，其名称及注释如下：

nsih.txt：uboot 编译的文件，默认为 g4418 开发板配置文件；

nsih-1G8b-800M.txt：ibox4418 1GB DDR3 的配置文件；
nsih-1G16b-800M.txt：g4418 开发板配置文件；
nsih-2G8b-800M.txt：ibox4418 2GB DDR3 的配置文件；

默认 nsih.txt 和 nsih-1G16b-800M.txt 的内容是完全相同的，它针对 g4418 开发板。我们 在编译 uboot 时，只会将 nsih.txt 编译进去，另外三个文件并不会编译进去。当编译 g4418 开发板的映像时，我们按默认参数即可；当编译 ibox4418 的 1GB DDR3 配置套餐时，需要将 nsih-1G8b-800M.txt 替换成 nsih.txt，再编译 uboot 映像；当编译 ibox4418 的 2GB DDR3配置套餐时，需要将 nsih-2G8b-800M.txt 替换成 nsih.txt，再编译 uboot 映像。

2.3.3 编译 kernel

在 linux 源码目录下执行如下命令编译 linux 内核:

```
./mk  -k
```

编译完成后映像文件 boot.img会释放到 out/release 目录。

g4418 开发板和 gbox4418 卡片电脑的内核源码包完全相同。

2.3.4 编译 Linux QT 文件系统

在 linux qt 源码目录下执行如下命令编译linux qt映像文件：
./mk –b
编译完成后映像文件qt-rootfs.img会释放到out/release 目录。注意，g4418 开发板和 gbox4418 卡片电脑的源码包完全相同。

第 3 章 制作启动卡

制作启动卡（量产卡）的过程，就是将映像 ubootpak.bin 烧写到 sd 卡（TF 卡）中。

3.1 Ubuntu 下制作启动卡

过程描述：准备一张SD卡，通过gparted工具将前面保留100多MB空间，后面格式化为FAT32分区；运行脚本s5p4418-sdmmc.sh制作启动卡

具体步骤：

第一步 :准备一张不小于 2GB 的 TF 卡 ,并插到装有 ubuntu 操作系统的 PC 机上；

第二步：删除 TF 卡的所有分区。

在 Linux 的终端窗口，使用 fdisk /dev/sdb 命令删除原来所有分区，sdb 为系统为 TF 卡分配的设备节点。注意，具体由节点名称而定，有可能是 sdc,sde 等。使用如下指令查询设备节点：

cat /proc/partitions

示例如下：

```
[root@david mass -production]# cat /proc/partitions
major minor  #blocks  name
   8        0   36700160 sda
   8        1     512000 sda1
   8        2   36187136 sda2
 253        0   34144256 dm-0
 253        1    2031616 dm-1
   8       16    3879936 sdb
   8       17    3875840 sdb1
[root@david mass -production]#
[root@david mass -production]# fdisk /dev/sdb

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): d
```

```
    Selected partition 1

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

    Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16:  设备或资源忙.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
    [root@rxs mass -production]#
```

输入 d，表示删除分区，输入 w 表示保存已经修改的分区信息。至此，原/dev/sdb1 被 删除。拔掉 TF 卡，再插入 PC 机上，查询设备节点：

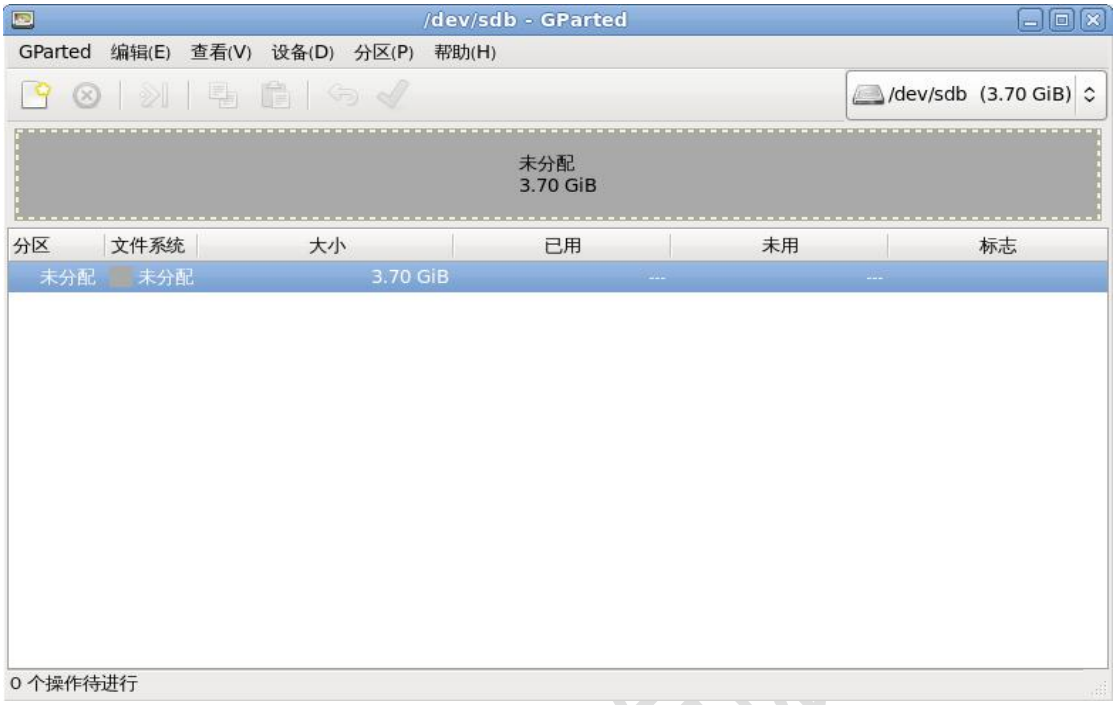
```
[root@rxs mass -production]# cat /proc/partitions
    major minor     #blocks     name

   8        0   36700160 sda
   8        1     512000 sda1
   8        2   36187136 sda2
 253        0   34144256 dm-0
 253        1    2031616 dm-1
   8       16    3879936 sdb
   [root@rxs mass -production]#
```

注意必须拔掉后再插入，否则仍然会提示存在/dev/sdb1 节点，会造成出错。

第三步：使用 gparted 工具给 TF 卡预留 256M 空间，用于存放 uboot 映像。 使用如下命令打开 TF 卡分区表：

gparted /dev/sdb



选择分区->新建，预留 256M 空间给 uboot，剩下的分区使用 fat32 格式，如下图所示

点击添加，选择菜单中的应用全部操作，完成 TF 卡的分区。

第四步：将 TF 卡剩余的空间格式化为 fat32 格式

sudo mkfs.vfat /dev/sdb1

第五步：进入映像生成目录，即 out/release 目录，执行如下指令烧写 ubootpak.bin 到 TF 卡：

烧写ubootpak.bin烧写命令：

sudo ./s5p4418-sdmmc.sh /dev/sdb ubootpak.bin

注意：这里/dev/sdb 为 TF 卡的节点，该节点为 linux 系统自动分配，也有可能为 sdc,sde 等，用户可查询节点名称后再执行上面的烧写脚。
这时，该 TF 卡就可以引导开发板启动 uboot 了。

注意：完成以上步骤，可以使用 TF 卡引导开发板启动 uboot；若需要 TF 卡具有升级功能，则需要拷贝相应的升级文件至 TF 卡中，具体操作方法，请参考本文档的 4.2.1 章节。

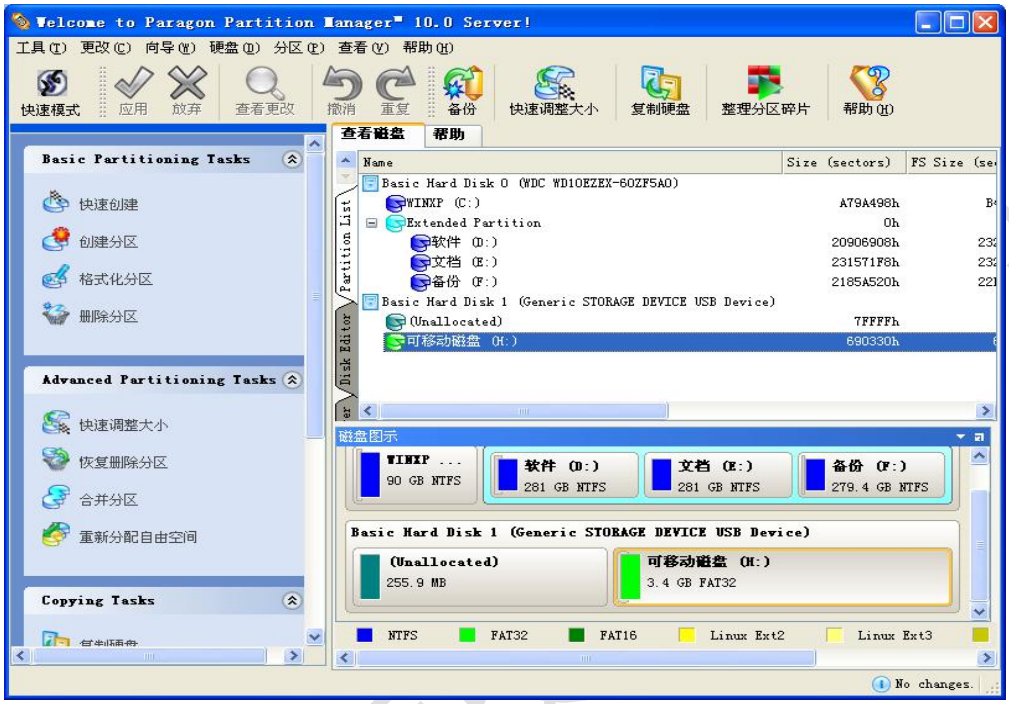
3.2 Windows 下制作启动卡

过程描述：将 TF 卡手动分区，预留其最前面的 256MB 空间存放 ubootpak.bin；通过烧写工具，将 ubootpak.bin 烧写到预留的空间中。

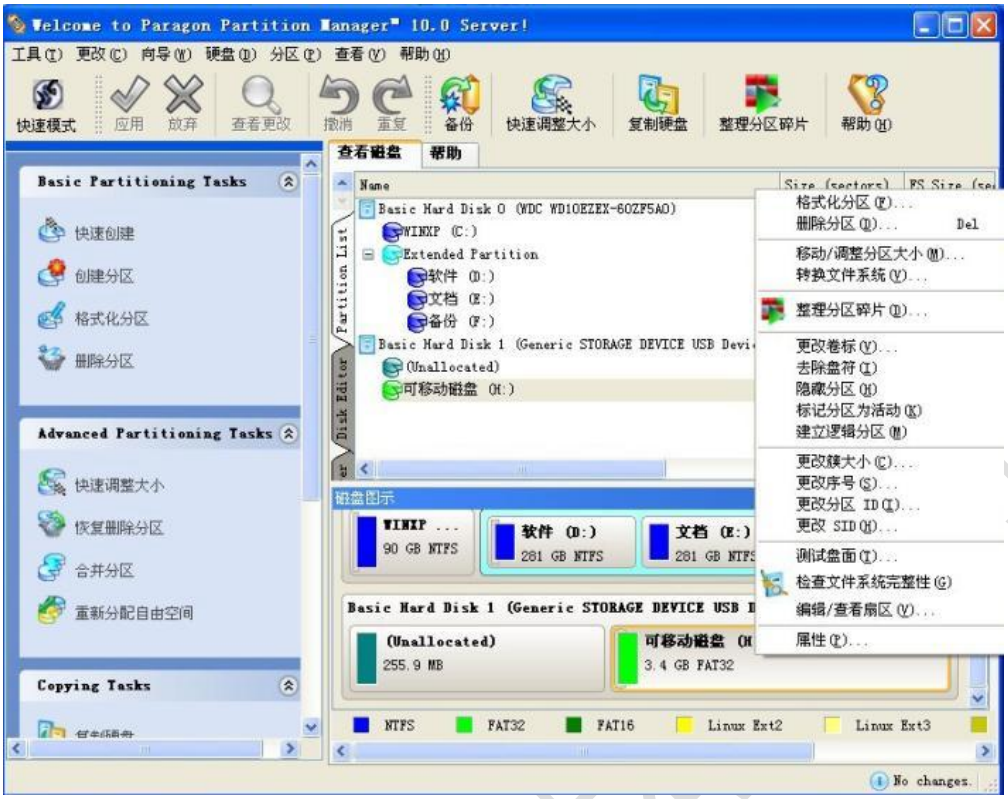
具体步骤：

第一步，准备一张容量不小于 2GB 的 TF 卡，通过读卡器连接到 Windows 操作系统的 PC 上。

第二步，进行分区；从开发资料中找到 PartitionManager.exe 工具并打开，界面如下：

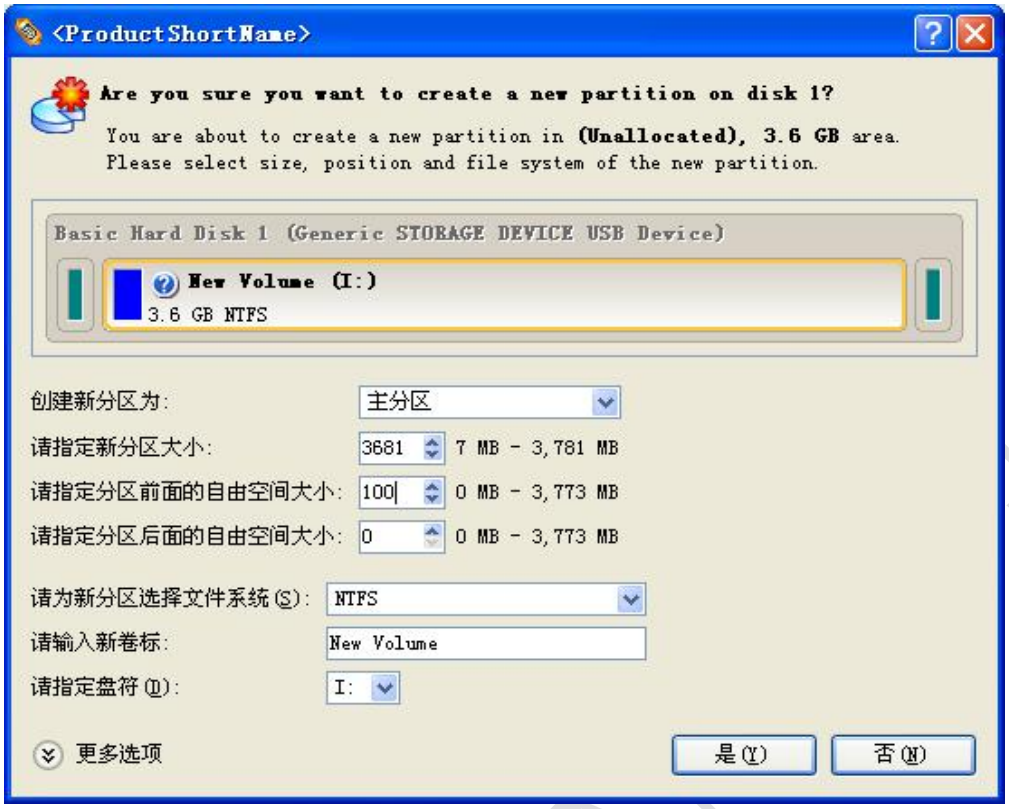


上图中 H 盘即为插入的 TF 卡分区，我们需要使用这个工具给 TF 卡预留一些空间，用于存放 ubootpak.bin。首先我们右键点击最下面图标的可移动磁盘(H:)，点击删除分区， 如下图：

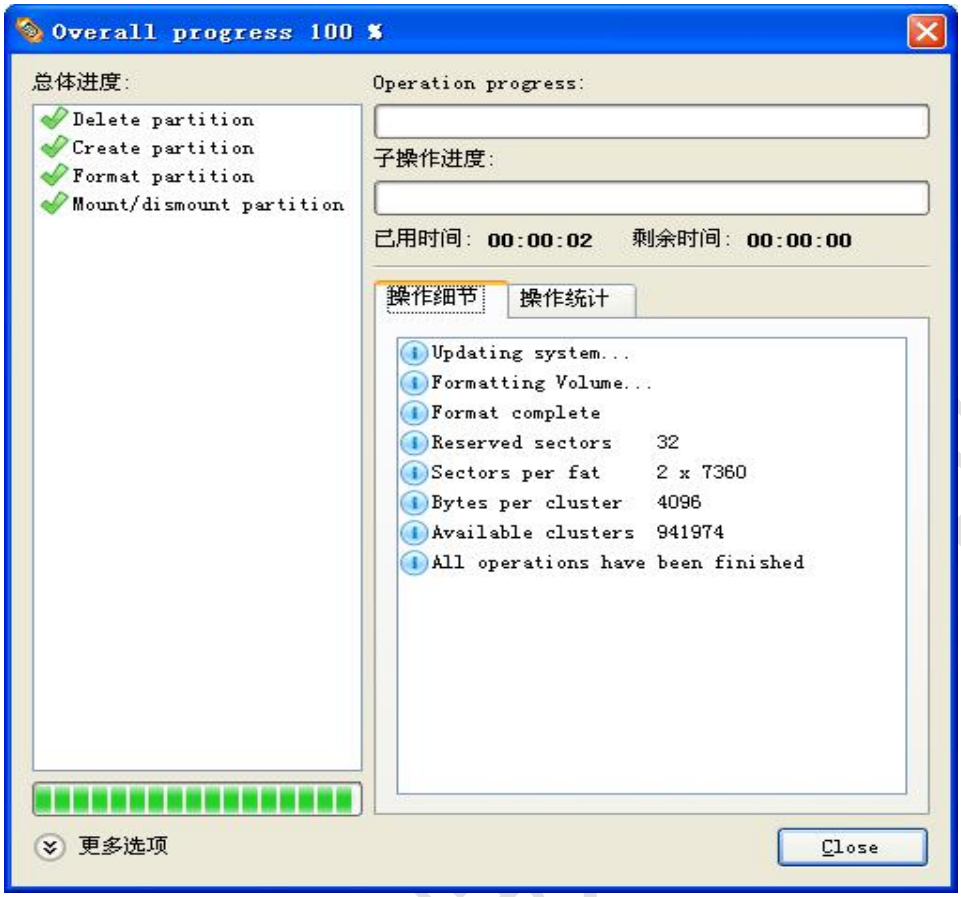


软件会弹出确认界面：

输入卷标名，勾上下次不再询问，点击是，即删除了原来的分区。这时，TF卡就只剩一个分区了。再次右键点击该分区，点击创建分区：



在请指定分区前面的自由空间大小一栏中填入我们需要预留的空间我们这里填 100M，留给 uboot 已经足够。为新分区选择文件系统中选择 FAT32，点击是，然后再到主界面快捷菜单栏点击应用即可。成功分区后提示如下：



第三步：分区成功，进行 ubootpak.bin 的烧写；打开 TF 卡烧写工具 V2.0，如下图：

进行以下操作：

在 SD/MMC Drive 一项中，选择 TF 的卷标；

点击 Browse 按钮，选择要烧写的 ubootpak.bin；接着点击 Add 按键，添加映像；

点击 START 按钮，进行烧写。烧写成功后，会弹出 Done 的对话框，提示烧写成功。

至此，该 TF 卡就可以引导开发板启动 uboot 了。

注意：完成以上步骤，可以使用 TF 卡引导开发板启动 uboot；若需要 TF 卡具有升级功能，则需要拷贝相应的升级文件至 TF 卡中，具体操作方法，请参考本文档的 4.2.1 章节。

第 4 章 烧写 Linux QT 映像文件

4.1 空板升级

空板，指的是核心板或者 gbox 的 emmc 中没有 ubootpak.bin 映像文件。

这种情况，对开发板或者 gbox 进行升级，需要使用启动卡（量产卡），即制作好的能作为启动卡的 TF 卡。

升级步骤：

1、制作启动卡

可以参考本文档的**第 3 章**，进行启动卡的制作。

2、在 TF 卡（启动卡）根目录下建立 g4418-qt 文件夹；

3、拷贝镜像文件：ubootpak.bin boot.img qt-rootfs.img 至 g4418-qt 目录下。

4、在 g4418-qt 目录下，创建环境变量默认配置文件 env.txt 不存在，则不更新系统环境变量

env.txt 文件内容配置示例:(需正确设置环境变量,在文件末尾需要保留一空行)

bootcmd=ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000

bootargs=root=/dev/mmcblk0p2 rw rootfstype=ext4 lcd=vs070cxn tp=gslx680-linux

5、系统上电，系统自动检测是否需要升级，等待即可

注意：空板情况下，除了上述使用启动卡（量产卡）进行升级方法外，也可以使用 USB 升

级；对于 USB 升级方法的描述，可查看相关的文档；

相对于使用 USB，用启动卡进行升级，方便简单，因此，我们建议客户采用**启动卡方式**进行空板升级。

4.2 正常升级

正常升级，指的是核心板或者 gbox 的 emmc 中已经烧写了 ubootpak.bin 映像文件。

4.2.1 使用 TF 卡脱机升级

升级步骤：

1、准备一张容量不小于 2GB 的 TF 卡；在 TF 根目录下建立 g4418-qt 文件夹；

2、拷贝镜像文件：ubootpak.bin boot.img qt-rootfs.img 至 g4418-qt 目录下。

3、在 g4418-qt 目录下，创建环境变量默认配置文件 env.txt 不存在，则不更新系统环境变量

env.txt 文件内容配置示例:(需正确设置环境变量,在文件末尾需要保留一空行)

bootcmd=ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000

bootargs=root=/dev/mmcblk0p2 rw rootfstype=ext4 lcd=vs070cxn tp=gslx680-linux

4、插入 TF 到开发板或 gbox，上电，系统自动检测是否需要升级，等待即可

4.2.2 Windows 下使用 fastboot 升级

具体步骤：

1、 fastboot 驱动安装（**若 PC 上已经有 fastboot 驱动，忽略此步骤**）

首次将开发板或 gbox 通过 otg 线与 PC 相连，板子上电后，电脑会提示安装 fastboot 驱动；可根据提示选择开发资料包中的 Fastboot_driver（Fastboot_driver.zip 解压后的文件夹），进行驱动的安装。

2、 安装 fastboot 工具（**若 PC 上已经安装，忽略此步骤**）

将开发资料包中的 fastboot.rar 文件解压到 windows 任意目录即完成安装。出于操作方便，一般解压到磁盘根目录，如 D 盘。

3、 进行映像更新

第一步，开发板或 gbox 接上 otg 线、串口线，连接到 PC；

第二步，PC 上打开串口终端，上电启动开发板或 gbox，在 uboot 三秒倒计时按下空格键进入 uboot 命令行，敲入指令 fastboot 或 fast，会列出分区表信息：

```
g4418# fast
Fastboot Partitions:
 mmc.2: ubootpak, img : 0x200, 0x7800
 mmc.2: 2ndboot, img : 0x200, 0x4000
```

mmc.2: bootloader, img : 0x8000, 0x70000

mmc.2: boot, fs : 0x00100000, 0x04000000

mmc.2: system, fs : 0x04100000, 0x2F200000

mmc.2: cache, fs : 0x33300000, 0x1AC00000

mmc.2: misc, fs : 0x4E000000, 0x00800000

mmc.2: recovery, fs : 0x4E900000, 0x01600000

mmc.2: userdata, fs : 0x50000000, 0x0

Support fstype : 2nd boot factory raw fat ext4 emmc nand ubi ubifs

Reserved part : partmap mem env cmd

DONE : Logo bmp 300 by 270 (3bpp), len=243054

DRAW : 0x47000000 -> 0x46000000

Load USB Driver : android

Core usb device tie configuration done

OTG cable Connected!

这时，开发板上会显示一幅等待升级的画面，并伴有相应的字符提示。

第三步，打开 cmd 命令行终端，使用如下指令更新映像：

fastboot flash ubootpak ubootpak.bin(指令一)
fastboot flash boot boot.img (指令二)

fastboot flash system qt-rootfs.img (指令三)

执行完以上指令，即可正常启动 linux qt 系统。每执行一条指令，在开发板或 gbox 的液

晶屏上都会有相应的界面提示，用户可以很清晰的观察升级的状态。

注意：

指令一等价于如下两条指令：

fastboot flash 2ndboot 2ndboot.bin

fastboot flash bootloader u-boot.bin

因此用户需要更新2ndboot或 uboot时，更新 ubootpak.bin 即可。

指令二理论上等价于如下两条指令：

fastboot flash kernel uImage

fastboot flash ramdisk ramdisk.img

因此用户需要更新内核或 ramdisk 时，更新 boot.img 即可。

第四步，设置 uboot 环境变量 bootargs，bootcmd：

bootcmd=ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000

bootargs=root=/dev/mmcblk0p2 rw rootfstype=ext4 lcd=vs070cxn tp=gslx680-linux

4.2.3 Ubuntu 下使用 fastboot 升级

具体步骤：

1、安装 fastboot（**若系统已经安装，忽略此步骤**）

执行如下指令安装 fastboot:

sudo apt-get install android-tools-fastboot

2、 安装 minicom，配置 minicom（**若系统已经安装，忽略此步骤**）

执行如下指令安装 minicom

sudo apt-get install minicom

配置 minicom，需要配置所使用的串口节点，波特率（115200），位数（8），停止位（1），无硬件流控，等；

sudo minicom –s

界面如下：

```
+-----[configuration]------+
| Filenames and paths
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

通过键盘的上下键盘选择到 Serial port setup，回车

```
+-----------------------------------------------------------------------+
```

```
| A -      Serial Device       : /dev/ttyWCH1                              |
| B - Lockfile Location    : /var/lock                                     |
| C -    Callin Program      :                                           |
| D -  Callout Program       :                                           |
| E -      Bps/Par/Bits        : 115200 8N1                                |
| F - Hardware Flow Control : No                                         |
| G - Software Flow Control : No                                         |
|                                                                        |
|     Change which setting?                                              |
+------------------------------------------------------------------------+
```

输入 A ，光标会停留到 Serial Device 的界面，将设备节点设置为/dev/ttyS0（这个节点需要与所使用的一致，此处只是一个示例）；

输入 F ，关闭硬流控；再回车，退出当前设置，回到上一界面，选择 Save setup as dfl，再选择 Exit 退 出设置。到此，minicom 安装、配置完成。

3、 新建 51-android.rules（**若系统已经存在此文件，忽略此步骤**）

新建 51-android.rules 文件，内容如下：

```
# adb protocol on passion (Nexus One)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e12",
MODE="0666",
```

```
OWNER="david"

# adb protocol on crespo/crespo4g (Nexus S)

SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e22",

MODE="0666",

OWNER="david"

# fastboot protocol on crespo/crespo4g (Nexus S)

SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e20",

MODE="0666",

OWNER="david"

# fastboot protocol on stingray/wingray (Xoom)

SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="708c",

MODE="0666",

OWNER="david"

# fastboot protocol on maguro/toro (Galaxy Nexus)

SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e30",

MODE="0666",

OWNER="david"

# fastboot protocol on g4418

SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="0002",

MODE="0666",
```

OWNER="david"

注意： OWNER 里面填的 " david" 务必换成用户 ubuntu 系统的用户名 。之后将 51-android.rules文件复制到/etc/udev/rules.d/ 目录下。

到此，就可以使用 fastboot更新映像。如果不建 51-android.rules这个文件，使用 fastboot更新时需要使用 root权限。

4、 进行映像更新

第一步，开发板或 gbox 接上 otg 线、串口线，连接到 PC；

第二步，PC 上打开串口终端，上电启动开发板或 gbox，在 uboot 三秒倒计时按下空格键进入 uboot 命令行，敲入指令 fastboot 或 fast，会列出分区表信息：

```
g4418# fast
Fastboot Partitions:
 mmc.2: ubootpak, img : 0x200, 0x7800
 mmc.2: 2ndboot, img : 0x200, 0x4000
 mmc.2: bootloader, img : 0x8000, 0x70000
 mmc.2: boot, fs : 0x00100000, 0x04000000
 mmc.2: system, fs : 0x04100000, 0x2F200000
 mmc.2: cache, fs : 0x33300000, 0x1AC00000
 mmc.2: misc, fs : 0x4E000000, 0x00800000
```

mmc.2: recovery, fs : 0x4E900000, 0x01600000

mmc.2: userdata, fs : 0x50000000, 0x0

Support fstype : 2nd boot factory raw fat ext4 emmc nand ubi ubifs

Reserved part : partmap mem env cmd

DONE : Logo bmp 300 by 270 (3bpp), len=243054

DRAW : 0x47000000 -> 0x46000000

Load USB Driver : android

Core usb device tie configuration done

OTG cable Connected!

这时，开发板上会显示一幅等待升级的画面，并伴有相应的字符提示。

第三步：另外开启一个 ubuntu 终端，使用如下指令更新映像：

fastboot flash ubootpak ubootpak.bin(指令一)
fastboot flash boot boot.img (指令二)

fastboot flash system qt-rootfs.img (指令三)

执行完以上指令，即可正常启动 linux qt 系统。每执行一条指令，在开发板或 gbox 的液晶屏上都会有相应的界面提示，用户可以很清晰的观察升级的状态。

注意：

指令一等价于如下两条指令：

fastboot flash 2ndboot 2ndboot.bin
fastboot flash bootloader u-boot.bin

因此用户需要更新2ndboot或 uboot时，更新 ubootpak.bin 即可。

指令二理论上等价于如下两条指令：

fastboot flash kernel uImage

fastboot flash ramdisk ramdisk.img

因此用户需要更新内核或 ramdisk 时，更新 boot.img 即可。

第四步，设置 uboot 环境变量 bootargs，bootcmd：

bootcmd=ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000

bootargs=root=/dev/mmcblk0p2 rw rootfstype=ext4 lcd=vs070cxn

tp=gslx680-linux

4.2.4 Uboot 环境变量设置

针对 android，linux，ubuntu 三种不同性质的操作系统，环境变量会有所不同，它们表
现在 bootargs 和 bootcmd 两个变量上。
例如，linux qt系统的环境变量为：

bootcmd " ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000"

bootargs " root=/dev/mmcblk0p2 rw rootfstype=ext4 lcd=vs070cxn

tp=gslx680-linux "

针对同一种操作系统如 android 或 linux 或 ubuntu， 如果对应不同的外设，如 LCD，触摸屏等，则它只表现在 bootargs 这个环境变量上。

本章节仅讲述 linux qt 系统的环境变量设置，无论外设如何变化，变量 bootcmd 都不会

改变，只需做如下设置即可：

setenv bootcmd " ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000"

save

以下为显示屏不同的设置参数：

7寸高清屏（1024x600）

setenv bootargs "lcd=vs070cxn tp=gslx680-linux "

save

VGA-1024x768:

setenv bootargs "lcd=vga-1024x768 "

save

VGA-1280x1024:

setenv bootargs "lcd=vga-1280x1024 "

save

VGA-1920x1080:

setenv bootargs "lcd=vga-1920x1080 "

save

7 寸 mipi 屏（1024x600）

setenv bootargs "lcd=wy070ml "

save

LVDS 11.6 寸屏（1366x768）

setenv bootargs "lcd=b116xtn04 "

save

5.5 寸 MIPI 屏（720x1280）

setenv bootargs "lcd=nst550"

save

第 5 章 Linux QT 测试程序

在 g4418 开发板上开发的QT测试软件基本上可以测试开发板的所有硬件功能，它在产品量产，程序开发上有很大的参考价值。开发板或gbox在安装了linux系统后，上电启动默认是进入测试界面，左右滑动，或者用鼠标滑动可切换测试的硬件。

测试软件可以测试发光二极管、蜂鸣器、按键、背光、AD转换、音频、触摸屏、串口、网卡、TF卡、U盘、休眠、重启、关机等；

根据客户需求，可提供此测试软件(qttest)的源码。

其它没有描述的硬件测试还包括：

录音 ,红外接收 ,RTC ,OTG USB ,PCIE 3G/4G ,GPS ,NFC ,WIFI, 蓝牙 ,HDMI ,MIPI/LVDS

屏，扩展 IO，等。

对于以上的测试，可与我们联系，获得相应支持。

第 6 章 使用 ramdisk 文件系统

默认的，已经将 ramdisk 文件系统打包到 linux 内核中，也就是说我们烧写的 boot.img 映像文件，其实里面已经存在 ramdisk 文件系统了。编译完 android 文件系统后，可以在 out/target/product/drone2/boot 目录下看到 ,里面存在 debug-ramdisk.img，我们可以通过修改 uboot 启动变量实现挂载 ramdisk。

boot$ ls

battery.bmp debug-ramdisk.img logo.bmp ramdisk-recovery.img root.img.gz uImage update.bmp

在 uboot 命令行下，依次输入如下三条指令：

ext4load mmc 2:1 0x48000000 uImage;

ext4load mmc 2:1 0x49000000 debug-ramdisk.img;

bootm 0x48000000

启动后，将会挂载 ramdisk文件系统

使用 ramdisk 系统，可以方便平时的调试。

第 7 章 产品线介绍

7.1 核心板系列

G4418 (主控为三星 4418)

G6818 (主控为三星 6818)

G210 (主控为三星 210)

M9 核心板 (主控为高通 8916)

7.2 开发板系列

G4418 开发板 (主控为三星 4418)

G6818 开发板 (主控为三星 6818)

G210 开发板 (主控为三星 210)

M9 开发板 (主控为高通 8916)

7.3 卡片电脑系列

G4418 卡片电脑 (主控为三星 4418)

G6818 卡片电脑 (主控为三星 6818)

G3188 卡片电脑 (主控为瑞芯微 3188)

说明：产品详细规格，以及更多其他产品请关注葡萄雨技术官方网站或与我们联系。