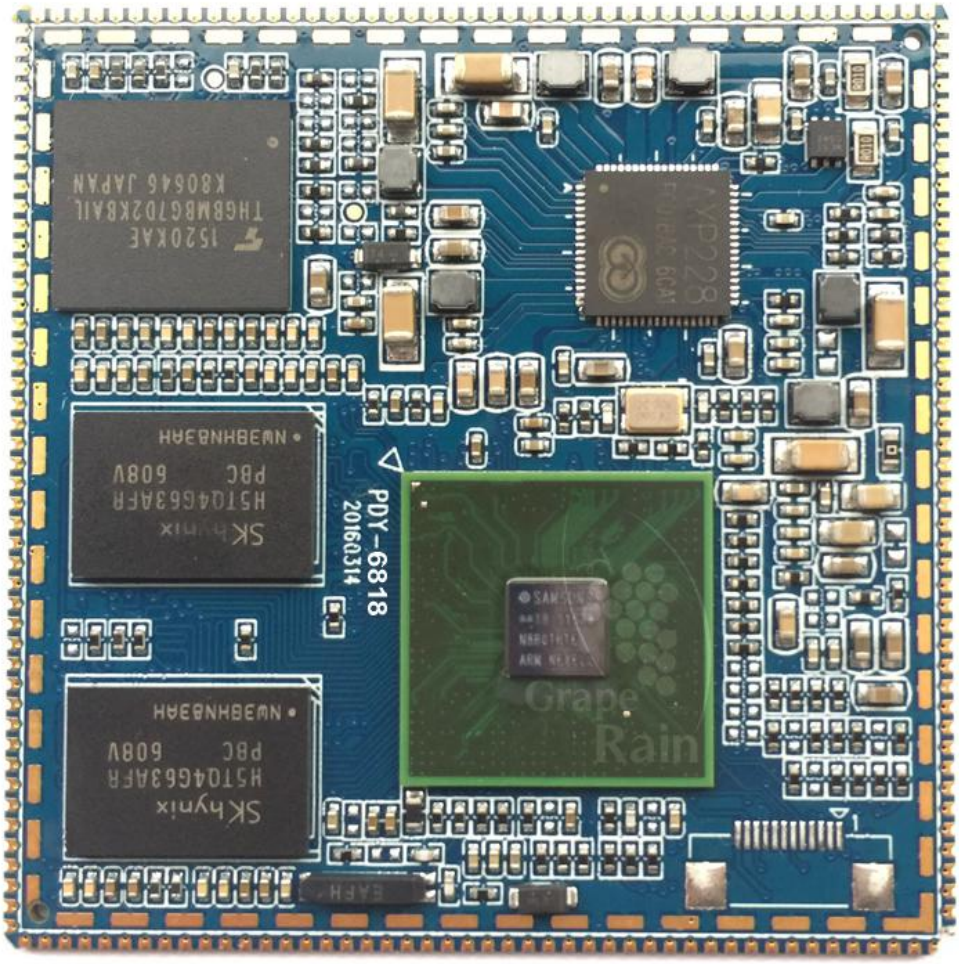
G6818 Android Platform User Guide



Shenzhen Graperain Technology Co., Ltd.

http://www.graperain.com/

Copyright Statement

Attention:

The manuals of development platform on sell will be updated from time to time, please download the latest manual from website www.graperain.com or contact our company sales representative, there would be no further notice.

Release notes

Version	Release Date	Author	Description
Rev.01	2016-4-22	David	Revision

Technical Support

Any questions about the manuals, you can call our landline or email us.

Website: http://www.graperain.com
Landline: +86 755 23025312
E-mail: supports@graperain.com

Sales and service network

Shenzhen Graperain Technology Co., Ltd.
Website: http://www.graperain.com
Landline: +86 755 23025312
E-mail: sales@graperain.com
Address: Building D, Huafeng Tech. & Innov. Park Baoan Wisdom Valley, Xixiang, Baoan Dist.Shenzhen, Guangdong. Post code 518101.

Contents

Chapter 1 Build Android Platform Environment

Please install Linux operating system in PC which could fully play PC performance as compiling Android source could ask for high requirements in PC hardware. We use Ubuntu14.04 64bit system in this chapter. Please keep the same with our version.

1.1 Install Ubuntu Through U-Disk

Install tools:

- One U-disk bigger than 2GB
- Linuxlive usb creater software, download: http: //www.linuxliveusb.com/
- Ubuntu 14.04 system, download: http://www.Ubuntu.com/download/desktop/

Install method:

Step1: ISO file with downloaded Ubuntu. Download linuxlive usb creater and install it.

Step2: Insert U-disk, open usb creator, setting it according to the software prompts. First, select installation disk, and find recognized U-disk; second, get Ubuntu image file; third, default; fourth, select file in hidden U-disk and format FAT32 it; fifth, click lightening icon to start installation, till done installation as following picture shows.

Step3: Restart computer, enter into BIOS setup menu, and take U-disk booting.
Desktop computer press DEL, while notebook press F2, some F10 log-in. Save when done setting and exit.
Step4: Restart system, and it shows Ubuntu installation interface, select language, and Go on;
Step5: Select install, and Go on;
Step6: Select language, and Go on;
Step7: Configure network, and take upgrade or not, or done installation and upgrade then.
Step8: Select something else, Go on, and ； separate two zones for Ubuntu, “一个 /” and”一个 /home”. Zones can be rebuit or format. It depends on requirements.
Step9: Setup area, take yours;
Step10: Select Keyboard;
Step11: Login user name and password, and done setup. Go on and install it directly. Restart when done installation, and enter into Ubuntu operating system.

1.2 Install Dependent Package of Android Source Code

State: All document based on Ubuntu14.06 64bits operating system.
Dependent package software package and 64bit System patches:

- GIT
- JDK 1.7
- git-core，gnupg，flex，bison，gperf，libsdl-dev，libesd0-dev，libwxgtk2.6-dev，libwxgtk2.8-dev，build-essential，zip，curl，libncurses5-dev，zlib1g-dev，libxml2-utils，genromfs，lsb-core，libc6-dev-i386，g++-multilib，lib32z1-dev，lib32ncurses5-dev，u-boot-tools，android-tools-fastboot,Texinfo

Use following command and install software package needed**:**
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install git-core gnupg flex bison gperf libsdl-dev libesd0-dev libwxgtk2.6-dev libwxgtk2.8-dev build-essential zip curl libncurses5-dev zlib1g-dev libxml2-utils genromfs lsb-core libc6-dev-i386 g++-multilib lib32z1-dev lib32ncurses5-dev u-boot-tools android-tools-fastboot Texinfo

Above software packages, we would suggest you install them one by one. This way could find which one failed clearly.

Manual Install JDK:

Steps of manual install JDK1.7 in Ubuntu:
Step1: execute the following commands:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java7-set-default
```

Step2: restart the system
Step3: check jave install version currently
root@david:/usr/lib/jvm# java -version

Until now, jdk1.7 has been installed successfully.

1.3 Install Cross-complier Tool

The cross-complier tool has been integrated into source code package yet, no need manual installation. Its path:(Android source code)
prebuilts/gcc/linux-x86/arm/arm-eabi-4.8

1.4 Specify GCC Cross-complier

When install latest GCC cross-complier in Ubuntu system, the version is exceed 4.4. Check GG version with the following command:

gcc –version

Maybe show you following information:

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

gcc (Ubuntu/Linaro 4.6.3-1Ubuntu5) 4.8.3

Copyright © 2011 Free Software Foundation, Inc.

Above information shows version is 4.8, it will be hint. That caused by new GCC version error, you could find solution in internet. If you do not want to modify those error, just lower your version into 4.4 will be ok.

Solution:

Install 4.4 version:

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

Done installation, and then start upgrading 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++

Check the version one more time and done upgrading:

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.

Chapter 2 Compiling Android Source Code Package

G6818 development board and G6818 single board computer is with Emmc memory chip default.

State: Take ordinary permissions when compiling image. There are three images needed only when done compiling. They are **ubootpak.bin，boot.img，system.img**

- Ubootpak.bin: bootloader which used to guide kernel, and it includes 2ndboot.bin，nsih and u-boot.bin. Packaged them into one file for debugging and updating new image.
- boot.img：it includes kernel uImage and ramdisk.
- system.img：android system image

State: Upgrading boot.img, that means upgrading uImage and ramdisk at same time.

2.1 Install Android Source Code Package

Copy android source code package from cloud storage into user's catalogue, and android 5.1 source code in cloud storage and name g6818_lollipop-20160428.tar.bz2. Notice: do not save this file into root catalogue of system file, if you do, there will be administration authority problem.
Sample: do following command in user's authority:

```
cp  yourcdromdir/source/ g6818_ lollipop -2060428.tar.bz2 ~/
cd
tar  xvf  g6818_ lollipop -20160428.tar.bz2
```

Now all Android file saved in the extracted directory. Till here all Android source code package installation done well.
State: Source code names could be different as its data issued, subject to cloud storage. This source code package supports G6818 development board and G6818 single board computer both.

2.2 Analysis Compiled Script

State: Kinds of source code version is different but principle same. And its script details subject to its relevant source code package, here it used to be analysis realizing mechanism.

Compiling script file MK content and annotation as following:

```
#!/bin/bash

#
# JAVA PATH
#
export PATH=/usr/lib/jvm/java-7-oracle/bin:$PATH

#
# Some Directories
#
BS_DIR_TOP=$(cd `dirname $0` ; pwd)
BS_DIR_RELEASE=${BS_DIR_TOP}/out/release
```

```
BS_DIR_TARGET=${BS_DIR_TOP}/out/target/product/G6818/
BS_DIR_UBOOT=${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
# State uboot and kernel cross-compiling tools
BS_CROSS_TOOLCHAIN_BOOTLOADER=${BS_DIR_TOP}/prebuilts/gcc/linux-x86/arm/arm
-eabi-4.8/bin/arm-eabi-
BS_CROSS_TOOLCHAIN_KERNEL=${BS_DIR_TOP}/prebuilts/gcc/linux-x86/arm/arm-eabi-4.
8/bin/arm-eabi-

#
# Target Config
# Specified configuration files for uboot, kernel and file system.
BS_CONFIG_BOOTLOADER_UBOOT=g6818_config
BS_CONFIG_KERNEL=g6818_defconfig
BS_CONFIG_FILESYSTEM=PRODUCT-g6818-userdebug
BS_CONFIT_BUILDROOT=g6818_defconfig

# Set up compiling environment before compiling to insure compiling stably
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
}

#Copy ubootpak,bin into out/release catalogue when done compiling uboot.
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
}

# Compiling kernel, when it done, the kernel uImage will be copied into out/release catalogue
automatically.
# Package kernel uImae and ramdisk file into boot.img, and copy it into out/release catalogue
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
}


#  Compiling android file system
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


#   compiling linux+ qt file system
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 Compiling Source Code

2.3.1 Checking Compiling Assistance

Do following command to checking usage of mk script
./mk –h

2.3.2 Compiling uboot

Do following command and compiling uboot in Android source code catalogue. Release image file ubootpak.bin into out/release catalogue when compiling done.

./mk -u

State: You have to compiling **2ndboot.bin, nsih** and **u-boot.bin three files** When compiling uboot default. Here we packaged this three files into one file and names **ubootpak.bin**So please careless **2ndboot.bin, nsih** and **u-boot.bin** three files when debugging normally.

Notice:

There are differences between G6818 development board and G6818 single board computer uboot. Default G6818 development board is of 16bit DDR3, and G6818 single board computer is of 8Bit DDR3.

Save following files in root catalogue of uboot source code package, and their names and annotation as following:

nsih.txt：uboot compiling file, default it as G6818 development board configuration file；

nsih-1G8b-800M.txt： the configuration file of 6818 SBC 1GB DDR3；

nsih-1G16b-800M.txt：the configuration file G6818 development board；

nsih-2G8b-800M.txt：the configuration file of G6818 SBC 2GB DDR3;

Identical content for nsih.txt and nsih-1G16b-800M.txt default, which subject to G6818 development board. When we compiling uboot, only nish.txt in, not other three files. When compiling G6818 development board image, just follow its default parameter; When compiling G6818 single board computer of 1GB DDR3 configuration set, just replace nsih-1G8b-800M.txt into nsih.txt and then compiling uboot image; When compiling G6818 single board computerof 2GB DDR3 configuration set, just replace nsih-2G8b-800M.txt into nsih.txt, and then compiling uboot image.

2.3.3 Compiling Kernel

Do following command compiling android kernel in Android source code catalogue:

./mk -k

Done compiling image, boot.img will be released into out/release catalogue

Same kernel source code package for G6818 development board and G6818 single board computer.

2.3.4 Compiling Android File System

First, please check if there is out catalogue in android root catalogue. At first compiling, cancel this whole out catalogue. Do following command and compiling android image file in android source code catalogue:

./mk -s

Done compiling image, the image file will be released into out/release catalogue

Notice, same source code package for G6818 development board and G6818 single board computer.

2.3.5 Generate boot.img

Boot.img file will be generated when compiling kernel and android file system.

Chapter 3 Make Boot Card

This chapter introduces the process of making boot card in bulk. In other way, program image ubootpak.bin into sd card (TF card).

3.1 Make Boot Card in Ubuntu

Process Description:
Prepare a SD card, reserve 100MB+ room in the front by the gparted tool, and format the back as FAT32 partition; run the script s5p6818-sdmc.sh to make the boot card.

Specified steps:

Step 1: Preparing a TF card more than 2GB and plug into PC with Ubuntu operating system.
Step 2: Delete all partition in TF card
In a Linux Terminal window, use the fdisk /dev/SDB command to delete all partitions. SDB is the system the device node assigned to TF card. Note that depending on the node name, and possibly SDC, SDE etc. Use the following command queries the device node:

cat /proc/partitions

Example：

```
[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: devices are busy.

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]#

Enter d means delete partitions. Enter w means save partition information which has been modified. At this point, the original /dev/sdb1 was deleted. Remove TF card, then insert the PC machine, query the device node:

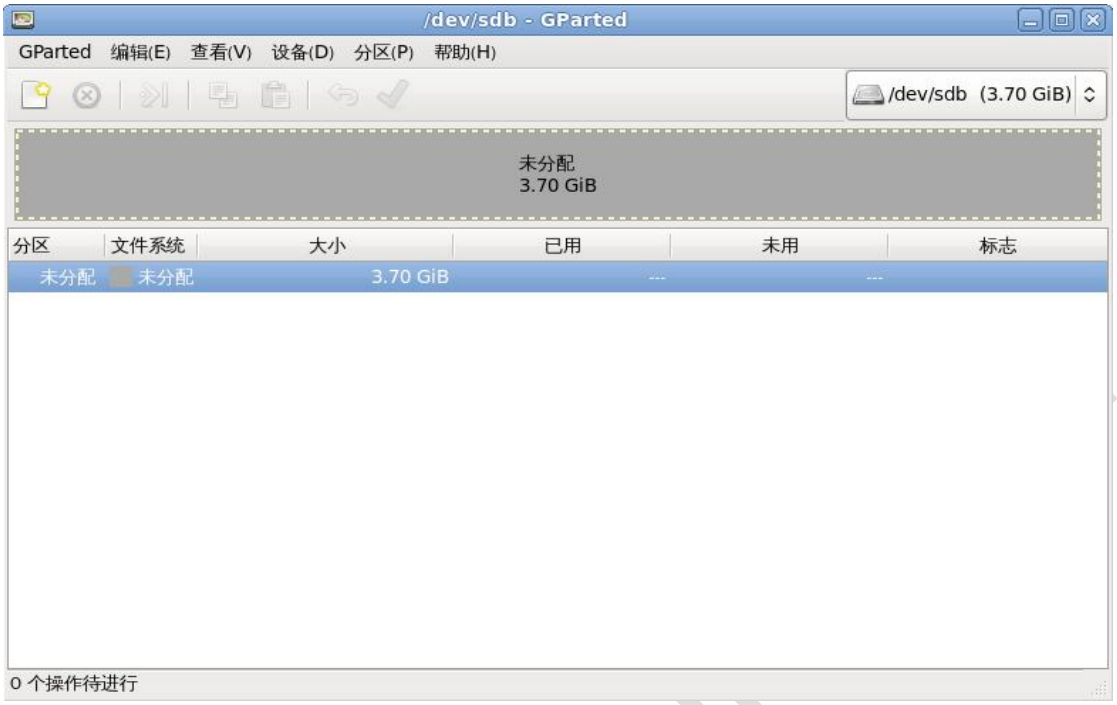
```
[root@rxs mass -production]# cat /proc/partitions
major minor  #blocks  name

   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]#
```

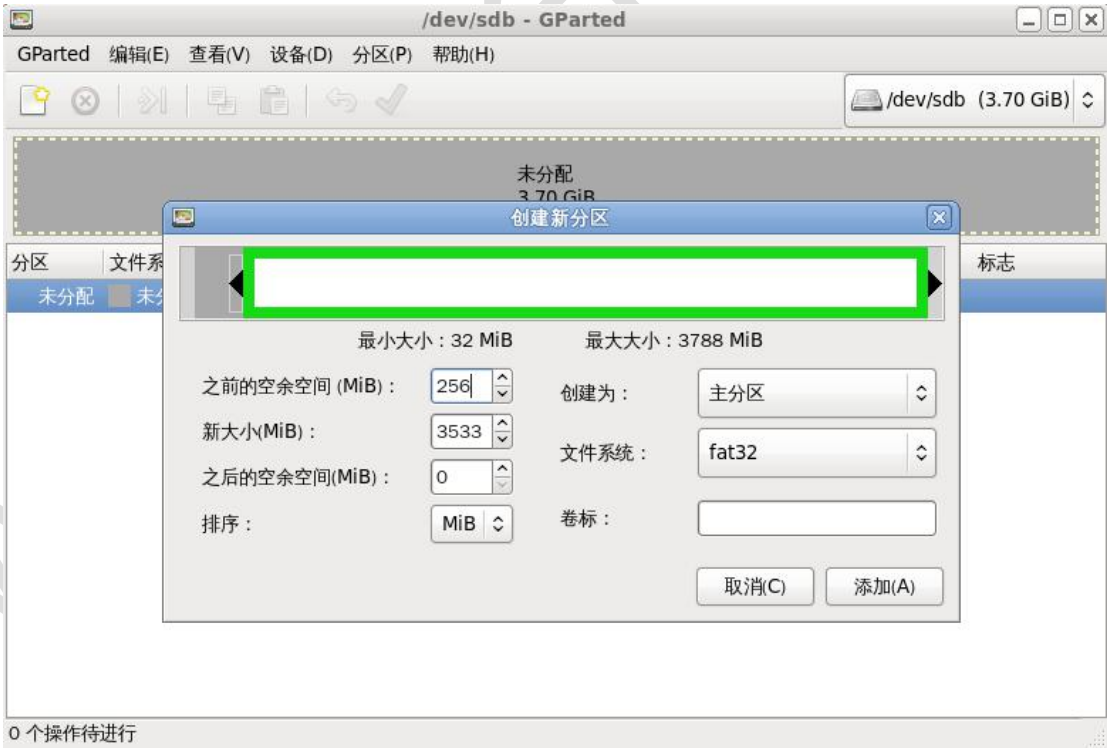
Note that you must remove the TF card then insert, otherwise it would indicate the node /dev/sdb1, errors may occur.

Step3: Take gparted tool to reserved 256M zone for TF card which used to save uboot image. Do following command to open TF card partition list:

gparted /dev/sdb



Select partition-> new, reserve 256M for uboot, the remaining partition that uses the FAT32 format, as shown in the following picture:



Click 添加(A), and select all application operating, done TF partition.

Step4: Formatting TF card rest zone into fat32

```
sudo    mkfs.vfat    /dev/sdb1
```

Stpe5: Enter into image and generate out/release, do following command and compiling ubootpak.bin into TF card:

Program ubootpak.bin：

```
sudo ./s5p6818-sdmmc.sh /dev/sdb ubootpak.bin
```

NOTICE:

1. The /dev/sdb here is the node of TF card, which is distributed by the Linux system automatically, maybe the sdc, sde and so on, users can run the above program pin after inquiring the device node name.
Now, the TF card can lead the development board to start uboot.

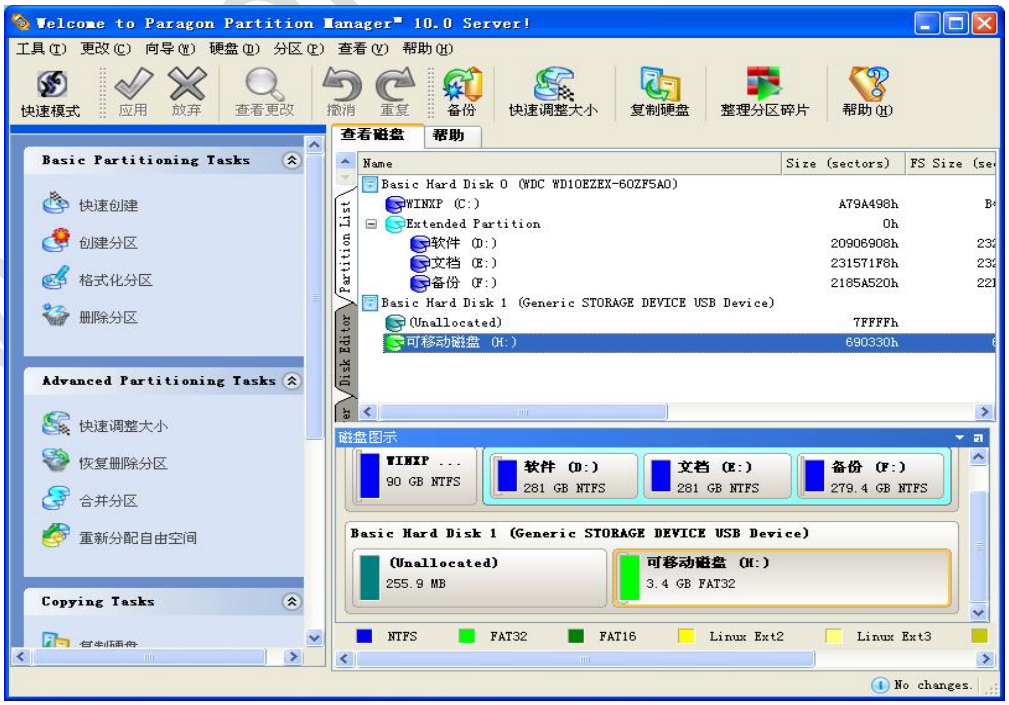
2. Complete the above steps, then we can use TF card and lead the development board to start the uboot; if the upgrade function is needed with the TF card, then we should copy the relevant upgrade files to TF card, the specific steps are referred to the Chapter 4.2.1 in this file.

3.2 Make Boot Card in Windows

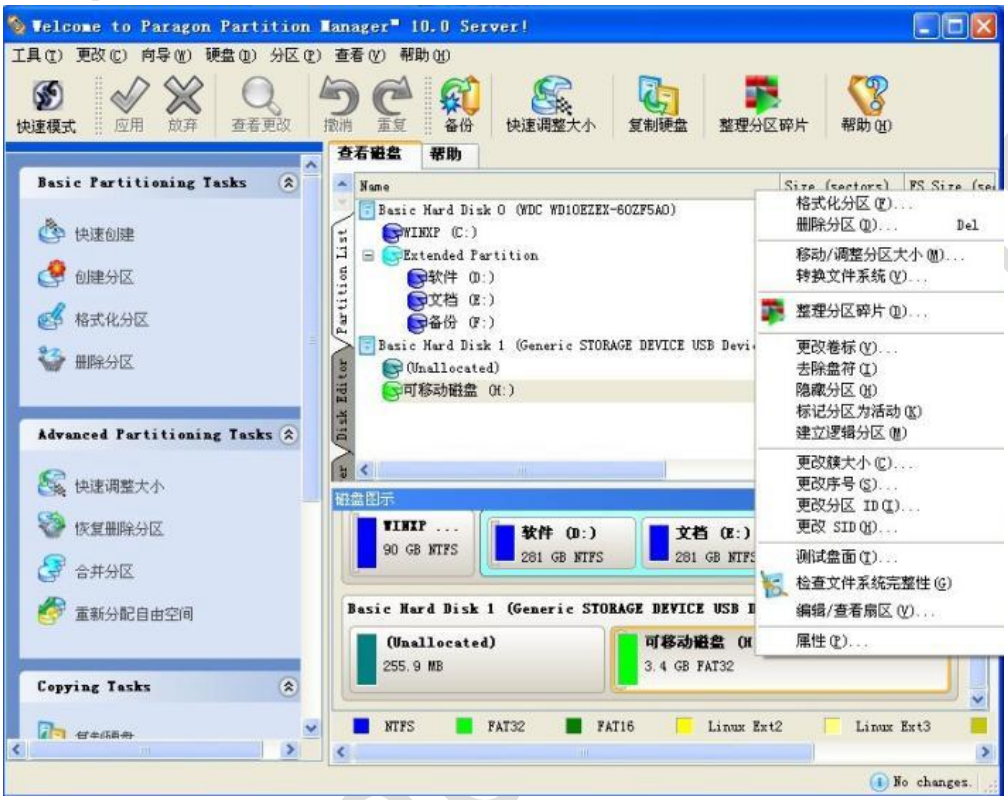
General: Manually partition the TF card, reserve the front 256MB space for store the ubootpak.bin; by using the programming tool, program the ubootpak.bin to the reserved space.

Specified Steps:

Step 1, prepare a TF card capacity more than 2GB, connect it to a Windows OS PC by card reader.
Step 2, enter the partition; find the PartitionManager.exe tool in the development files provided by Graperain and open it, the interface is as follows:

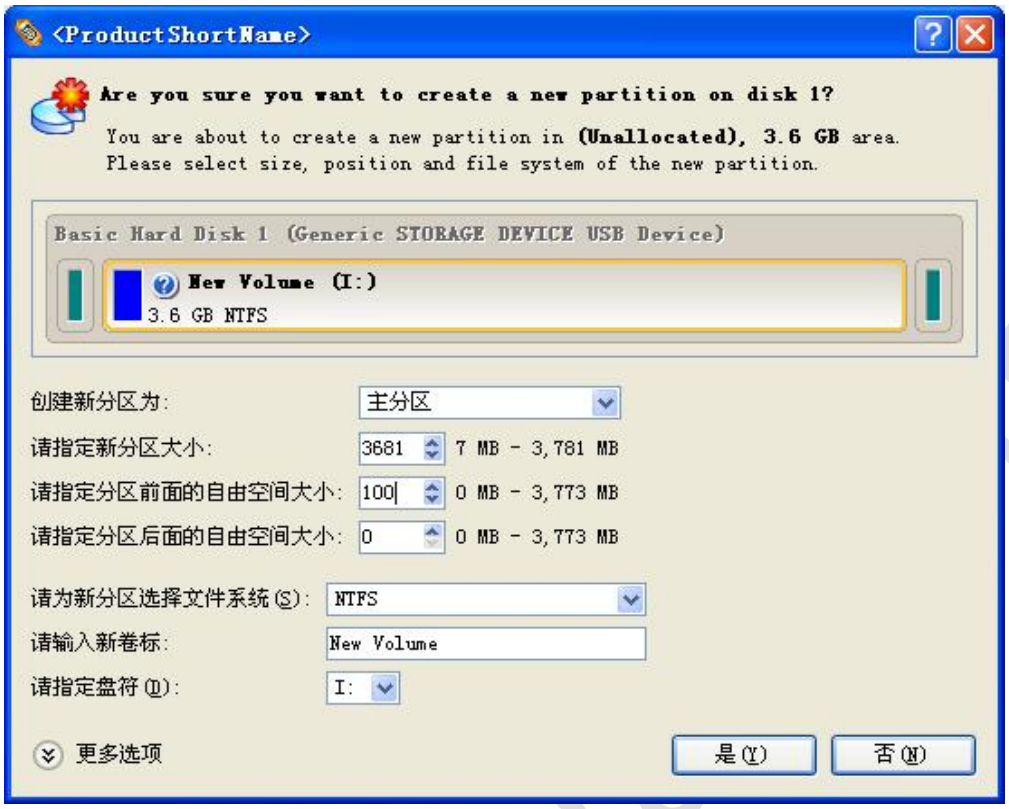


The 可移动磁盘（H：）（disk H:）in the pic above is the partition of the inserted TF card, we need to use this tool to reserve some space for the TF card to store ubootpak.bin. First of all, right click the icon 可移动磁盘（H：）（disk H:）, then click 删除分区（D）… (Delete partition), as shown in the below pic:

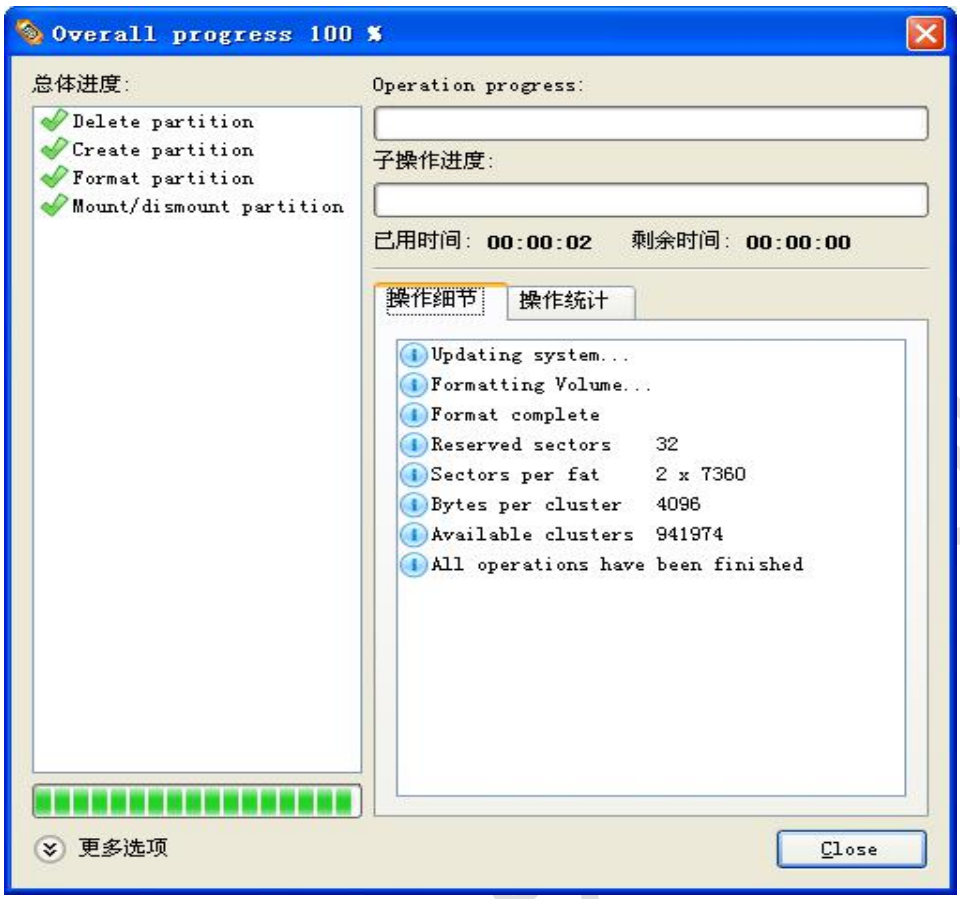


Confirm interface will pop-up

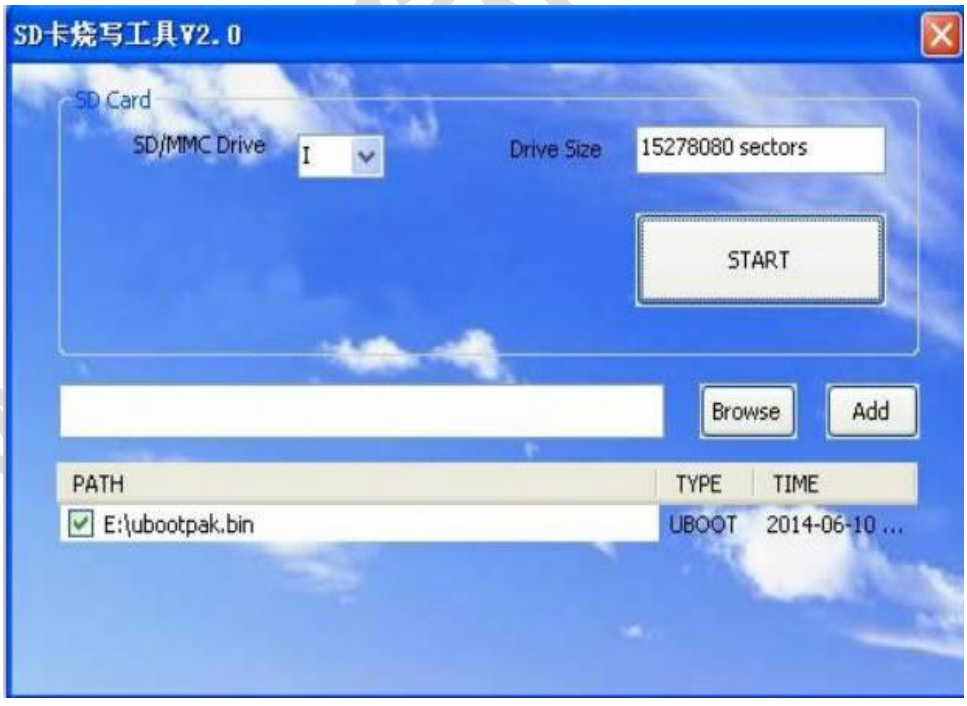
Enter the volume label name, check the 下次不再询问（D）(do not ask next time), click 是（Y）, delete the partition now. There is only one partition left in SD card. Right click it again to create a new partition:



Please fill in 100 in the bar 请指定分区前面的自由空间大小（Please specify a capacity before partition）, here 100MB is enough for uboot. Please choose FAT32 in the bar 为新分区选择文件系统(S) and click 是（Y）,then return to the main interface and click 应用（Apply）in the shortcut menu. After a successful partition it will show as follows:



Step 3, partition successful, proceed to the ubootpak.bin programming; open the SD 卡烧写工具 V2.0（SD card programming tool v2.0）:



And do as follows:

Choose the TF volume label in bar SD/MMC Drive;
Click Browse button, choose the ubootpak.bin; then click Add button to add the image;
Click START button to program. When the programming is successful, a dialog box of Done will show up, indicating programming successful.

So far, the TF card can lead the development board to launch uboot.

Notice: When the above steps are done, we can use the TF card to lead the development board to launch uboot; if the upgrade function is needed with the TF card, then please copy the relevant files for upgrading to the TF card, the specified steps are referred to Chapter 4.2.1 in this file.

Chapter4 Compiling Android Image Files

4.1 Naked Board Upgrade

Naked board is that SOM or single board computer which no ubootpak.bin image file in emmc.
In this way, if you want to upgrade development board or single board computer, boot card will be required, that is TF boot card.

Upgrade Steps:
Step1: Make boot card
Please reference Chapter3 about way of making boot card.
Step2: Set up G6818-android file in TF card root catalogue.
Step3: Copy image files ubootpak.bin boot.img system.img into G6818-android catalogue. Update it if you copied userdata.img，cache.img，recovery.img.

Step4: It won't upgrade system environment variable if env.txt does not exist in G6818-android.
env.txt sample of file content configuration: (keep one null string in end of file if it ask for set up environment variable)

bootcmd=ext4load mmc 2:1 0x48000000 uImage;ext4load mmc 2:1 0x49000000 root.img.gz;bootm 0x48000000
bootargs=lcd=vs070cxn tp=gslx680

Step 5: Power on system, and it will test if ask for upgrade or not, just waiting.

Notice: Besides naked board upgrade as above making boot card methods. USB could be upgraded too. Please refer to relevant file of USB upgrade;
Boot card is simpler than USB upgrade, we would suggest you take boot card way to upgrade naked board.

4.2 Normal Upgrade

Normal upgrade means ubootpak.bin IMAGE have been compiled into eMMC in SOM or SBC already.

4.2.1 Use TF Card Off-line Upgrade

Steps：
Step1: Prepare a 2GB more TF card; set up G6818-android file in TF root catalogue;
Step2: Copy image files ubootpak.bin, boot.img and system.img into G6818-android. Update it if you copied userdata.img, cache.img, recovery.img.

Step3: It won't upgrade system environment variable if env.txt does not exist in G6818-android
env.txt sample of file content configuration: (keep one null string in end of file if it ask for set up

environment variable)

bootcmd=ext4load mmc 2:1 0x48000000 uImage;ext4load mmc 2:1 0x49000000 root.img.gz;bootm 0x48000000

bootargs=lcd=vs070cxn tp=gslx680

Step4: Insert TF card into development board or single board computer, power-on, and the system will test it ask for upgrade or not, just waiting.

4.2.2 Use Fastboot to Upgrade in Windows

Steps:

Step1: Install fastboot drive (If PC has fastboot drive, please ignore this step)

First connect development board or single board computer with PC through OTG line, power on, and PC will hint install fastboot drive; following hints to select data package from Fastboot_driver (which have been released well), install its drive then.

Step2: Install fastboot tool (If PC has fastboot drive, please ignore this step)

Release fastboot.rar file into windows catalogue will be done. Generally, we would suggest you release it into disk root catalogue, such as D disk.

Step3: Update Image

First, connect development board or single board computer with PC through OTG line or serial cable.

Second, Open serial terminal in PC, power on and boot development board or single board computer. Press space key in three seconds countdown in uboot and enter into uboot command, input fastboot or fast, it will list out partition information.

```
G6818# 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!
```

By now, screen will shows waiting for the upgrade, and hint.

Step3: Open cmd command terminal, and upgrade image by following commands:

fastboot flash ubootpak ubootpak.bin (Command 1)

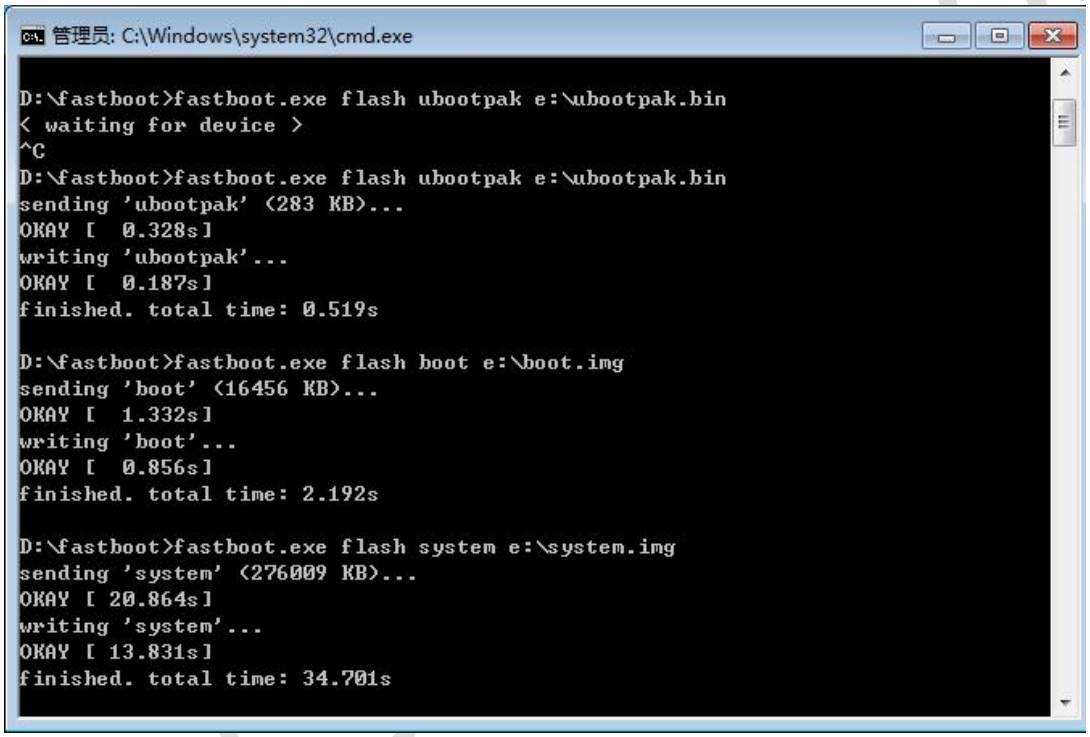
fastboot flash boot boot.img (Command 2)

fastboot flash system system.img (Command 3)

fastboot flash cache cache.bin (Command 4)

fastboot flash userdata userdata.img (Command 5)

Normal operation interface is as follows:



Done above commands, android system will boot normally. It will be hint for every commands done from development board or single board computer screen in interface. User could check its upgrade status clearly.

Notice:

Command 1 means following two commands:

fastboot flash 2ndboot 2ndboot.bin

fastboot flash bootloader u-boot.bin

Upgrade ubootpak.bin when users need to upgrade 2ndboot or uboot.

Command 2 means following two commands:

fastboot flash kernel uImage

fastboot flash ramdisk ramdisk.img

Upgrade boot.img when users need to upgrade kernal or ramdisk.

4.2.3 Upgrade by Fastboot in Ubuntu

Steps:

Step1: Install fastboot (If PC has it, please ignore this step).
Do following command to install fastboot:
sudo apt-get install android-tools-fastboot

Step2: Install minicom, configuration minicom (If PC has it, please ignore this step).
Do following command to install minicom
sudo apt-get install minicom

Configurate minicom, that means configurate all practical serials nodes. Such as baud rate (115200), digits (8), stop bits（1）, no hardware flow control and so on.
sudo minicom –s
Interface as following:

```
+-----[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          |
+----------------------------+
```

Select Serial port setup by keyboard up and down, and click Enter.

```
+-----------------------------------------------------------------------+
| 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?                                            |
+-----------------------------------------------------------------------+
```

Input A, and cursor will stop in Serial Device, and set up /dev/ttyS0 as device node (this node has to accordance with used one, here just example);
Input F, close hardware flow control; Enter, and exit current setting, back last interface, select Save setup as dfl, and Exit. By now, done installation of minicom and configuration.

Step3: New set up 51-android.rules (If PC has it, please ignore this step here).
New set up 51-android file, and content as following:

```
# 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"
```

Notice: Change OWNER="david" to your own user name in Ubuntu system. And then copy 51-android.rules into /etc/udev/rules.d/.

By now, it is ok upgrade image by fastboot. If no 51-android.rules file, fastboot upgrade ask for root privilege.

Step4: Upgrade image:
First, connect development board or single board computer with PC through OTG, serials line;
Second, open PC serial terminal, power on and boot development board or single board computer. Press space key in three seconds countdown in uboot, and enter into uboot command, input fastboot or fast, and following partition information will be list out:

```
G6818# 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!

Now screen will shows waiting for upgrade interface, and hint.

Third, open the other Ubuntu terminal, and do following commands to upgrade image:

```
fastboot flash ubootpak ubootpak.bin ( command 1)
fastboot flash boot boot.img ( command 2)
fastboot flash system system.img (command 3)
fastboot flash cache cache.bin (command 4)
fastboot flash userdata userdata.img (command 5)
```

Done above commands, android system will boot normally. It will be hint details when every commands done, users could check its upgrade status clearly.

Notice:
Command 1 means following two commands:

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

If users want to upgrade 2ndboot or uboot, just upgrade ubootpak.bin will be ok.

Command 2 means following two commands:

```
fastboot flash kernel uImage
fastboot flash ramdisk ramdisk.img
```

If users want to upgrade kernal or ramdisk, just upgrade boot.img will be ok.

4.2.4 Set Up Uboot Environment Variable

There are different environment variable subject to Android, Linux + QT, Ubuntu operating system, which shows on bootargs and bootcmd.

Such as, environment variable of android system is:

bootcmd "ext4load mmc 2:1 0x48000000 uImage;ext4load mmc 2:1 0x49000000 root.img.gz;bootm 0x48000000"

bootargs "lcd=vs070cxn tp=gslx680"

Its environment variable shows on bootargs only if you want to connect with various peripherals such as LCD, touchscreen etc subject to one specific operating system, such as Android, or Linux + QT, or Ubuntu.
This chapter shows Android environment variable set up. No matter which peripherals change, its bootcmd will not changed, just set up as following:

setenv bootcmd " ext4load mmc 2:1 0x48000000 uImage;ext4load mmc 2:1 0x49000000 root.img.gz;bootm

0x48000000"

save

Screen set up parameters as following:
7inch HD（1024x600）:

setenv bootargs "lcd=vs070cxn tp=gslx680"

save

VGA-1024x768:

setenv bootargs "lcd=vga-1024x768 tp=gslx680"

save

VGA-1280x1024:

setenv bootargs "lcd=vga-1280x1024 tp=gslx680"

save

VGA-1920x1080:

setenv bootargs "lcd=vga-1920x1080 tp= gslx680"

save

7inch mini Screen（1024x600）

setenv bootargs "lcd=wy070ml tp=gslx680"

save

LVDS 11.6 inch screen（1366x768）

setenv bootargs "lcd=b116xtn04 "

save

5.5inch mini screen（720x1280）
setenv bootargs "lcd=nst550"
save

Chapter 5 Android Test Program

It could be test all hardware functions in G6818 development board Android testing software, which is a great reference value when bulk production. Test APP interface and click android test, enter into testing, using mouse or cursor switching to test hardware.

The tests includes: LCD screen, TP, LED, Beep, Keys, Battery, ADC, Serial, TF card, U disk, Gsensor, Camera, Ethernet, WIFI etc.

We could provide testing software source code based on users requirements.

Other hardware testing not includes as following, we would provide technology supports about them.

Recording，infrare receiver，RTC，OTG USB，PCIE 3G/4G，GPS，NFC，bluetooth，HDMI，MIPI/LVDS screen，extend IO，and so on。

Chapter 6 Product Portfolio

6.1 System on Modules

G4418 System On Module (SoC is Samsung S5P4418)
G6818 System On Module (SoC is Samsung S5P6818)
G210 System On Module (SoC is Samsung S5PV210)
M9 System On Module (SoC is Qualcomm MSM8916)

6.2 Development Boards

G4418 Development Board (SoC is Samsung S5P4418)
G6818 Development Board (SoC is Samsung S5P4418)
G210 Development Board (SoC is Samsung S5PV210)
M9 Development Board (SoC is Qualcomm MSM8916)

6.3 Single Board Computers

G4418 Single Board Computer (SoC is Samsung S5P4418)
G6818 Single Board Computer (SoC is Samsung S5P6818)
G3288 Single Board Computer (SoC is Rockchip RK3288)

Instructions: For more detailed specifications and other products, please pay attention to www.graperain.com or contact us directly.