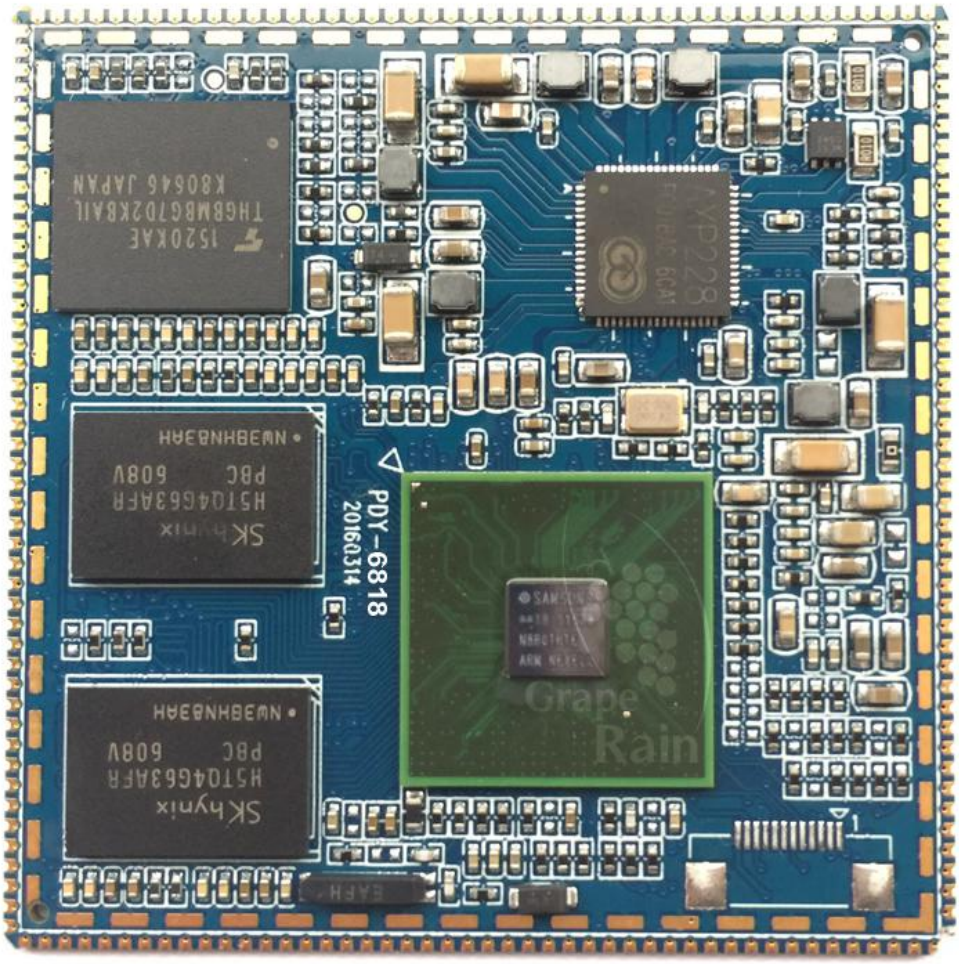
G6818 Linux QT User Manual



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

Copyright Statement....2
1.1 Use U Disk to Install Ubuntu....6
1.2 Install Relative Dependency Package....7
1.3 Install Cross Compile Tool-chain....8
1.4 Specifying GCC Cross-compiler....8
Chapter 2 Build Linux QT Source Code Package....10
2.1 Install Linux QT Source Code Package....10
2.2 Compiling Script Analysis....10
2.3 Compile Source Code....15
2.3.1 Check Compile Help....15
2.3.2 Compile uboot....15
2.3.3 Compile Kernel....16
2.3.4 Compile Linux QT File System....16
Chapter 3 Make Boot Card....17
3.1 Make boot card in Ubuntu....17
3.2 Make boot card in Windows....20
Chapter 4 Program Linux QT Image File....25
4.1 Naked Board Upgrade....25
4.2 Normal Upgrade....25
4.2.1 Use TF Card Off-line Upgrade....25
4.2.2 Use Fastboot to Upgrade in Windows....26
4.2.3 Upgrade by Fastboot in Ubuntu....27
4.2.4 Set Up Uboot Environment Variable....31
Chapter 5 Linux QT test procedures....33
Chapter 6 Use Ramdisk File System....34
Chapter 7 Product Portfolio....35
7.1 System on Modules....35
7.2 Development Boards....35
7.3 Single Board Computers....35

Chapter 1 Build Linux QT Development Environment

Linux QT development is not like Android which requires high standards for PC performance. Users can install Ubuntu OS in virtual machine to develop Linux.
For how to install virtual machine in Windows, how to install Ubuntu in virtual machine, users can Google it or check our file *Install Ubuntu in virtual machine*.
In the development, we use Ubuntu 14.04, 64 bit system. Users can keep the same version as ours.

1.1 Use U Disk to Install Ubuntu

Install tools:

- U disk bigger than 2G
- Linuxlive usb creater software, download：http://www.Linuxliveusb.com/
- Ubuntu 14.04, download：http://www.Ubuntu.com/download/desktop/

Install method:
Step 1. Downloaded ISO file of Ubuntu and Linuxlive usb creater and install.
Step 2. 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 Relative Dependency Package

Notice: All the development in this doc is based on **Ubuntu14.04** 64-bit system
Dependency package and 64-bit system patch:

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, Texinfo

Use the following commands to install software package:
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 Texinfo

Suggest install the above package one by one to find which one is not installed successfully.

1.3 Install Cross Compile Tool-chain

Cross Compile Tool-chain has been integrated source code package, no need to install manually.
Cross Compile Tool-chain path:（in Android source code）
prebuilts/gcc/Linux-x86/arm/arm-eabi-4.7

1.4 Specifying GCC Cross-compiler

When you install the latest GCC cross compiler on Ubuntu system, the version is newer than 4.4, use the following command to query GCC version:

gcc –version

Possible interface as follows:
david@Ubuntu-server:~$ gcc --version
gcc (Ubuntu/Linaro 4.6.3-1Ubuntu5) 4.8.3
Copyright © 2011 Free Software Foundation, Inc.

Users will find that version is 4.8, there will prompt some mistakes with this version. you can find all solutions online. If you won't want to search for them, you can choose install the version 4.4.

Solution:

Install 4.4 version

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

Finish installing, start downgrading 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 again, downgrading complete:

```
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 Build Linux QT Source Code Package

G6818 development board and G6818 single board computer is with Emmc memory chip default.
Notice: When compile image, use general authority. After compiling, only three images needed: **ubootpak.bin, boot.img, qt-rootfs.img.**

- ubootpak.bin: bootloader uses to lead Kernel. It includes 2ndboot.bin, nsih and u-boot.bin three files. Pack them into a file to debug and update images conveniently.
- boot.img：including uImage and ramdisk.img
- qt-rootfs.img：Linux qt system imagefile system image

Notice: If update **boot.img, uImage** and **ramdisk** will be updated simultaneously**.**

2.1 Install Linux QT Source Code Package

Copy Linux Source code Package from the web, put it into user directory. The name of Source code Package is *G6818_Linux-20160428.tar.bz2*. Don't put them into root directory to avoid management authority problem.
Example: execute commands in User Authority:

```
cp    yourcdromdir/source/ G6818_Linux-20160428.tar.bz2   ~/
cd
tar   xvf   G6818_Linux-20160428.tar.bz2
```

Decompress the file, 4 files in the directory:
uboot： store uboot source code；
kernel：store kernel source code；
buildroot：store Linux files system source code；
Linux：store uboot,kernel source code,some dependent library, function library and testing procedures etc..

Now, Linux QT source code package installed finished.

Notice:
1, Source code package name may differs from the date. The exact according to the web. The source code package supports G6818 Development Board and G6818 single board computer.
2, Uboot, kernel of Linux and Android are compatible. When using capacitive touch screen in Linux, only need to change bootargs in boot environmental variables in uboot. Details refer to uboot environmental variables settings chapter.

2.2 Compiling Script Analysis

Notice: Each version of source code in build script is nearly the same. Principles are the same. Specific script refer to the relative source code package. Here is to analysis Implementation Mechanism.

Compile script file mk 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
# claim uboot and kernel Cross Compile Tool-chain
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
#target uboot，kerel and file system config file

BS_CONFIG_BOOTLOADER_UBOOT=G6818_config
BS_CONFIG_KERNEL=G6818_defconfig
BS_CONFIG_FILESYSTEM=PRODUCT-G6818-userdebug
BS_CONFIT_BUILDROOT=G6818_defconfig

#before compiling, set compile environment to ensure reliably.
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
}

#compile uboot. After compiling, ubootpak.bin will be automatically copied to out/release directory.

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}/s5p6818-sdmmc.sh ${BS_DIR_RELEASE}

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

#compile uboot. After compiling, it will automatically copy kernel image uImage to out/release
directory.
#Pack kernel image uImage and ramdisk file system to boot.img，copy to out/release directory

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
}

# Compile 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
}

# Compile 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 telechiNotice 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 Compile Source Code

2.3.1 Check Compile Help

Run the following command to query how to use mk scripts:

```
./mk –h
```

2.3.2 Compile uboot

Run the following commands to compile uboot in Linux source code directory. Image file ubootpak.bin will be released into out/release directory after finish compiling:

```
./mk    -u
```

Notice: when programming uboot, users need to program **2ndboot.bin, nsih** and **u-boot.bin** three files. Here we package the three files into one file, and named ubootpak.bin. Therefore, when debugging, no longer need to pay attention to these three files 2ndboot.bin, nsih and u-boot.bin.

Notice: There is a bit difference of uboot between G6818 Development Board and G6818 single board computer.The standard DDR3 for G6818 Development Board is 16 BIT, while G6818 SBC is 8 BIT DDR3. DDR configuration is different.

There are some files in uboot source code package root directory, their name and notation as following:
nsih.txt：uboot compile file，G6818 Development Board default configuration file；
nsih-1G8b-800M.txt：G6818 SBC 1GB DDR3 configuration file；
nsih-1G16b-800M.txt：G6818 Development Board configuration file；
nsih-2G8b-800M.txt：G4418 SBC 2GB DDR3 configuration file；

The contents in nsih.txt and nsih-1G16b-800M.txt are exactly the same, it's for G6818 Development Board. When compile uboot, only nsih.txt will be built in. The other three files won't. When compile G6818 Development Board image, go by default parameters. When compile G6818 SBC with 1GB DDR3, change nsih-1G8b-800M.txt to nsih.txt, then compile uboot image. When compile G6818 SBC with 2GB DDR3, change nsih-2G8b-800M.txt to nsih.txt, then compile uboot image.

2.3.3 Compile Kernel

Performing the following commands to compile Linux kernel in Linux source code directory:

```
./mk    -k
```

After finish compiling, the kernel image boot.img will be released to out/release directory.

The Kernel source code package of G6818 Development Board and G6818 single board computer are the same.

2.3.4 Compile Linux QT File System

Execute the following commands to compile Linux QT image file in Linux qt source code directory:

```
./mk –b
```

After finish compiling, the kernel image qt-rootfs.img will be release to out/release directory.

The Kernel source code package of G6818 Development Board and G6818 single board computer are the same.

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 to delete partitions. Enter w to save partition information has been modified. Now, the original /dev/sdb1 was deleted. Remove TF card, then insert to PC machine, query the device node:

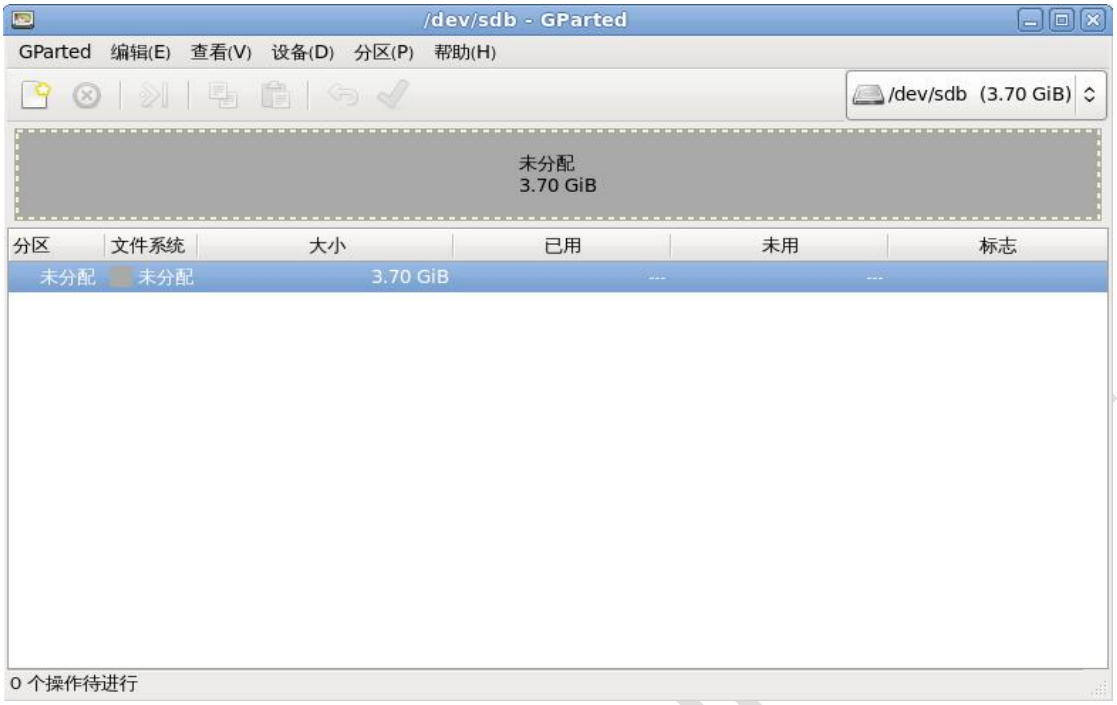
```
[root@rxs mass -production]# cat /proc/partitions
    major minor       #blocks       name

   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]#
```

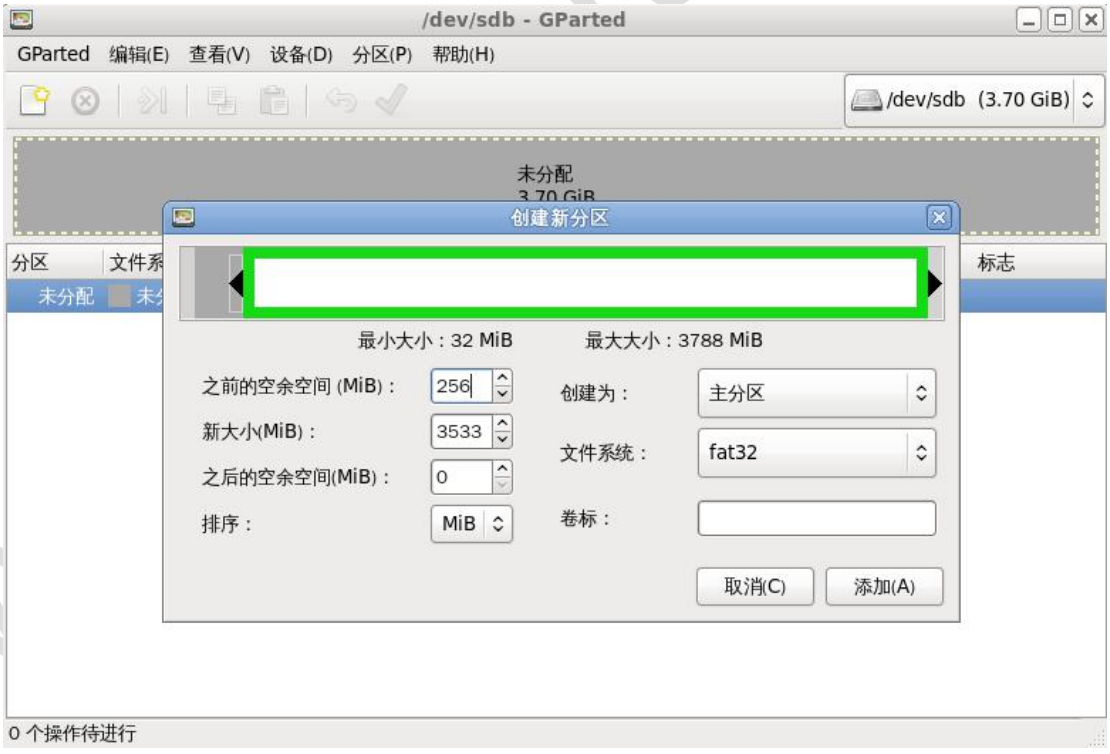
Note, users must insert the TF card again after remove it. Otherwise there would still prompt /dev/sdb1 node, cause errors.

Step 3: Reserve 256M space for TF card using gparted, to store the uboot image.
Open the TF card partitioned table using the following command:

```
    gparted /dev/sdb
```



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



Click Add, select all in the menu and complete the TF card partitions.

Step 4: Format TF card remaining space in fat32.

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

Step 5: Enter the image generate directory, which is out/release directory, perform the following instructions to burn ubootpak.bin to TF card:
ubootpak.bin program commands:

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

NOTICE:

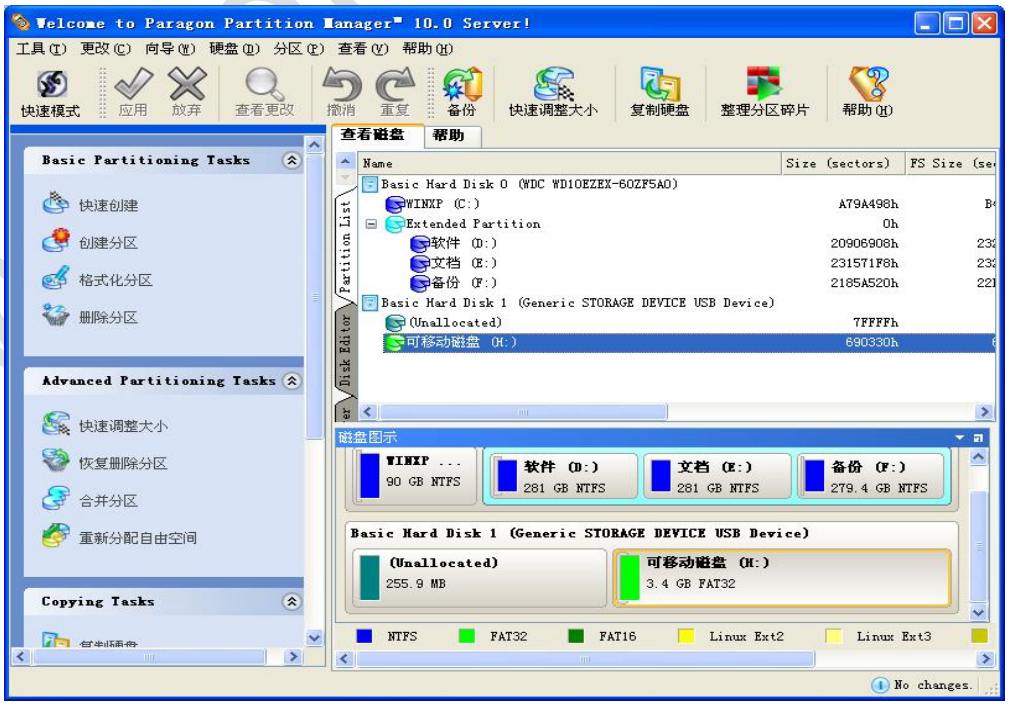
1. The /dev/sdb here is the node of TF card, which is distributed by the Linux system automatically, maybe the sdc, sde and so on, users can run the above program pin after inquiring the device node name.
Now, the TF card can lead the development board to start uboot.

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 Chapter 4.2.1.

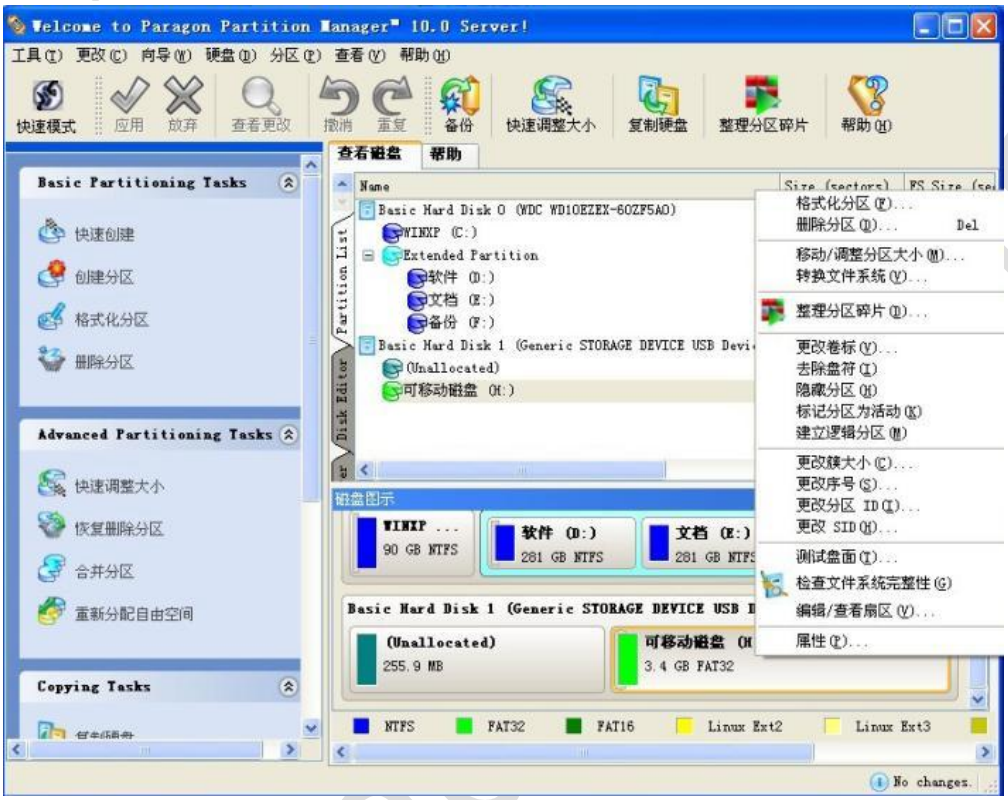
3.2 Make boot card in Windows

Procedure: prepare a SD/TF card, reserve more than 256MB space to store ubootpak.bin. Copy ubootpak.bin to reserved space by programming tool.

Step 1: Preparing a TF card more than 2GB, connect with Window operating system by card reader.
Step 2: Partition. Find and open PartitionManager.exe in the files, shown as below:

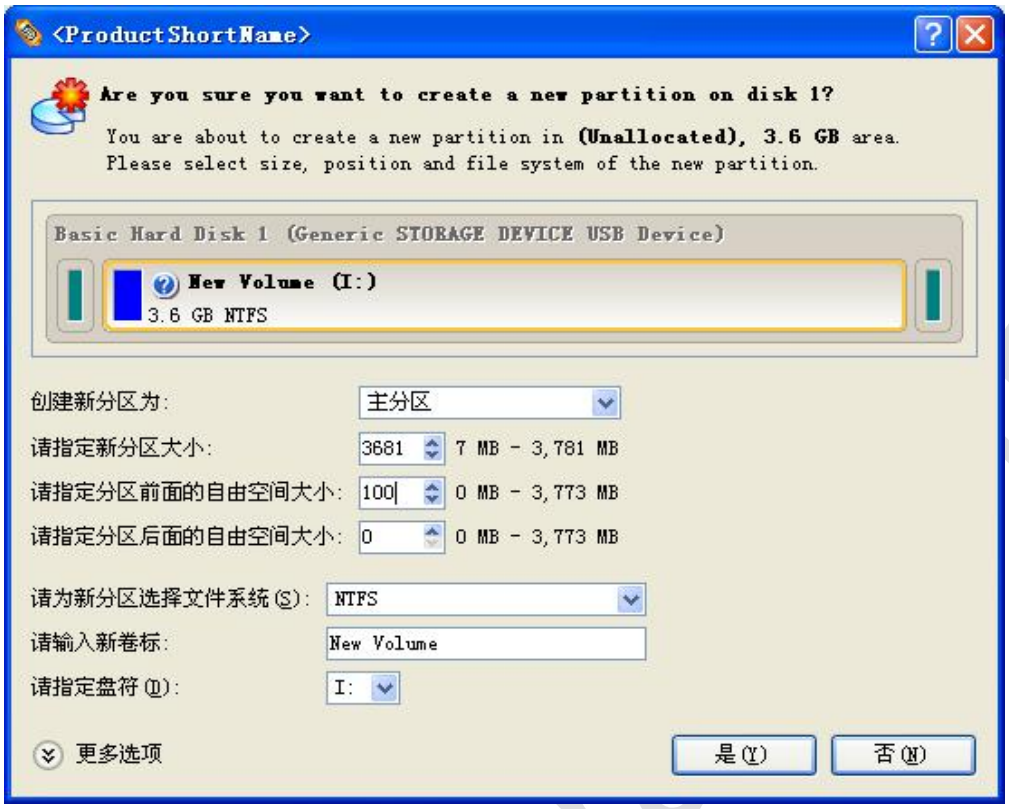


The 可移动磁盘（H：）（disk H:）in the pic above is the partition of the inserted TF card, we need to use this tool to reserve some space for the TF card to store ubootpak.bin. First of all, right click the icon 可移动磁盘（H：）（disk H:）, then click 删除分区（D）… (Delete partition), as shown in the below pic:

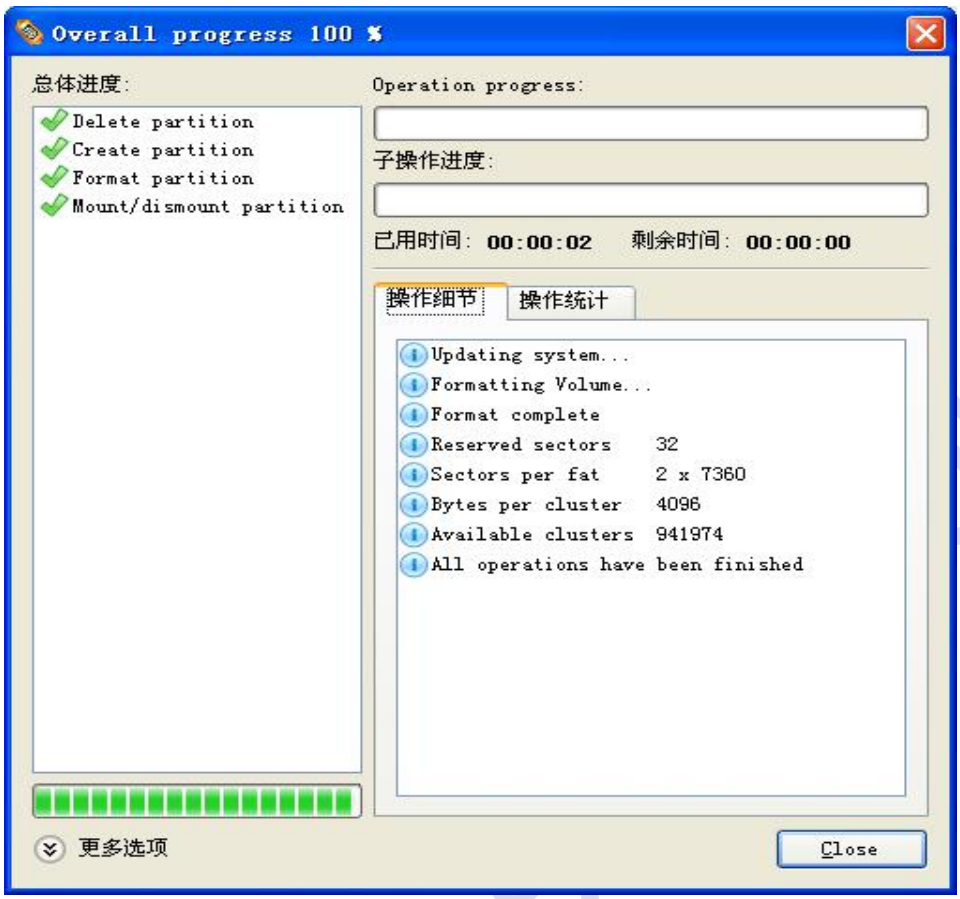


Software will pop up a confirmation:

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.

Chapter 4 Program Linux QT Image File

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. Build G6818-qt folder under the root directory in TF card(boot card);
Step3. Copy image ubootpak.bin boot.img qt-rootfs.img into G6818-qt catalog;
Step4. If there is no env.txt in directory G6818-qt, do not need to upgrade system environment variables
Please check the example of env.txt file contents:(There need to keep a blank line in the end of the file to set the environment variables correctly)
bootcmd=ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000
bootargs=root=/dev/mmcblk0p2 rw rootfstype=ext4 lcd=vs070cxn tp=gslx680-Linux

Step5.Start the board, system detects whether need to upgrade automatically, 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:
1. Prepare a TF card more than 2GB, Build G6818-qt folder under the root directory in TF card;
2. Copy images ubootpak.bin, boot.img, qt-rootfs.img into G6818-qt folder.
3. If there is no env.txt in directory G6818-qt, do not need to upgrade system environment variables
Please check the example of env.txt file contents:(There need to keep a blank line in the end of the file to set the environment variables correctly)
bootcmd=ext4load mmc 2:1 0x48000000 uImage;bootm 0x48000000

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

4, 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 line terminal, upgrade image with following command:

```
fastboot flash ubootpak ubootpak.bin (command 1)
fastboot flash boot boot.img (command 2)
fastboot flash system qt-rootfs.img (command 3)
```

Executing the above instructions, users can normally start the Linux qt system. Each execution of an instruction, the screen will prompt a corresponding interface, users can observe the status of the upgrade clearly.

Note：

Command 1 is equivalent to the following two commands:

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

So users just need to upgrade ubootpak.bin when need to upgrade 2ndboot and uboot.

Command 2 is equivalent to the following two commands:

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

So users just need to upgrade boot.img when they need to upgrade kernel and ramdisk.

Step 4: Set up uboot environmental variables 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 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).
Run the 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
The interface is as bellow:

```
                +-----[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 "↑""↓" on keyboard, then press "Enter"

```
  +---------------------------------------------------------------------+
    | A -       Serial Device          : /dev/ttyWCH1                                  |
    | B - Lockfile Location        : /var/lock                                  |
    | C -     Callin Program          :                                              |
    | D -   Callout Program         :                                              |
    | E -       BNotice/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).

Bulid a new 51-Android.rules 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/.

For now, users can use fastboot to upgrade image. Users need to use root permission when use fastboot to upgrade image if there is no 51-Android.rules.

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 qt-rootfs.img (command 3)
```

Above commands is executed, you can start the Linux+QT system normally. Executing each instruction and in the corresponding interface prompts on the LCD, users can clearly observe the status of the upgrade.

Note：

Command 1 is equivalent to the following two commands:

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

So users just need to upgrade ubootpak.bin when need to upgrade 2ndboot and uboot.

Command 2 is equivalent to the following two commands:

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

So users just need to upgrade boot.img when need to upgrade kernel and ramdisk.

Step 4:Set up uboot environmental variables 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 Set Up Uboot Environment Variable

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

For example, Linux+QT system environment variable is:

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

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

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 just introduce how to set up Linux+QT's environmental variables, for different peripheral devices, the bootcmd is same, just set up it as below:

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

save

The following are the parameter settings of different display:

7-inch HD Screen（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-inch mipi screen（1024x600）
setenv bootargs "lcd=wy070ml "

save

11.6-inch LVDS Screen（1366x768）
setenv bootargs "lcd=b116xtn04 "
save

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

Chapter 5 Linux QT test procedures

QT test software on G6818 Development Board can almost test all the hardware functions. It's significant reference in mass production and program developing. After installing Linux on Development Board or SBC, power on it can go into test interface. Slide left or right, or use a mouse can switch the hardware.

The test includes: LED, Beep, Keys, LCD light, ADC, Audio, TP, Serial, Ethernet, TF card, U disk, Sleep, Reboot, Poweroff.

According to customers' requirements, this test software qttest source code can be provided.

Other hardware test not described in this doc including: recording，IR receiver，RTC，OTG USB，PCIE 3G/4G，GNotice，NFC，WIFI, bluetooth，HDMI，MIPI/LVDS screen，extended IO，etc.
For the above tests, please contact us to get support.

Chapter 6 Use Ramdisk File System

The default configuration has packed ramdisk file system within the Linux kernel, that's to say, the boot.img we program, actually there has been the ramdisk file system. When compiling Android file system is completed, it can be seen under the directory: out/target/product/drone2/boot, there exists the debug-ramdisk.img, we can mount the ramdisk by modifying uboot start-up variables.

boot$ ls

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

Under the uboot command line, input the following three commands successively:

ext4load mmc 2:1 0x48000000 uImage;

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

bootm 0x48000000

After startup, it will mount the ramdisk file system, the successfully mounted interface.

Use ramdisk system can be more convenient to debug.

Chapter 7 Product Portfolio

7.1 System on Modules

G4418 SoM (SoC is Samsung S5P4418)
G6818 SoM (SoC is Samsung S5P6818)
G210 SoM (SoC is Samsung S5PV210)
M9 SoM (SoC is Qualcomm MSM8916)

7.2 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)

7.3 Single Board Computers

G4418 SBC (SoC is Samsung S5P4418)
G6818 SBC (SoC is Samsung S5P6818)
G3188 SBC (SoC is Rockchip RK3188)

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