**Modifying the android boot Image for Open Sourcing(Lollipop)**

The process below describes how to open source the kernel code. Blackburn devices in the field are based off of Intel build 503034. It is recommended that open sourcing be done based on boot image in 503034 build. This process will enable users to swap the source boot.img inside e.g 503034 build with a desired target boot.img.

**Note: This document describes steps for opensourcing kernel in Lollipop based images only.**

**Prerequisites:**

* **Following modules are needed**
	+ **split\_bootimg.pl** – a script to unpack boot image
	+ **mkbootimg** – a script to pack boot image(vendor/intel/support)
	+ **unpack-ramdisk** - unpack ramdisk
	+ **repack-ramdisk** - repack ramdisk
	+ **Iconic\_diffconfig** file
* **Perform following steps on the BlackBurn signed image:**
1. Download an Intel build e.g. 503034.
2. Extract the blackburn-wifi-qs-user-fastboot-xxx.zip from /flash\_files/build-user directory and remove the following line from flash.xml.

<command>

 <string>fastboot flash splashscreen $splash\_file</string>

 <timeout>120000</timeout>

 <retry>2</retry>

 <description>Flashing splashscreen</description>

 <mandatory>1</mandatory>

</command>

1. Flash the provided qs\_blankphone-user file on to the device.
2. Flash the blackburn-wifi-qs-user-fastboot-xxx.zip on to the device.
* **Misc Information about other modules**
1. This Iconic\_diffconfig file disables kernel signature generation, so that the boot image can power up on a rooted device. The picture below shows the content of Iconic\_diffconfig file.



1. Makefile\_kernel -Users can modify the Makefile, as per their environment.



**Procedure:**

1. Create a working directory e.g. kernel\_open\_source.
2. Extract the Boot.img file from the build 503034 into the working directory.
3. Copy the split\_bootimg.pl script into the working directory.
4. Unpack Boot.img using split\_bootimg.pl script.

perl split\_bootimg.pl boot.img

1. Extract the ramdisk

./unpack\_ramdisk boot.img-ramdisk.gz

Unpacking creates in the working directory which includes boot components ( ramdisk, kernel, second), as shown below.



1. **Creating the target Kernel boot.img**

Users would need to do a repo sync to the target build, which contains the target kernel that is desired to be used with the existing 503034 build.

1. Setup the android build environment
2. Copy “Makefile\_kernel” and iconic\_diffconfig into android root repo.



1. Build the kernel image.

 source build/envsetup.sh

 Lunch product-variant (e.g Lunch Blackburn-wifi-qs-userdebug)

 make –f <Makefile\_kernel >

The newly built target “kernel” image is available in “\out\target\product\iconic”

1. Replace old kernel image in the working directory created above with target kernel image

cp kernel\_open\_source\kernel \out\target\product\iconic \kernel

1. Repack the ramdisk from given directory.

./repack\_ramdisk / <ramdisk directory name> <output file name>

1. Pack target boot.img in the working directory. The cmdline parameters to be used for packing are copied from cmdline parameters when boot.img is unpacked.

./mkbootig –second <filename> --kernel <filename> –ramdisk <filename> --cmdline <filename> –output <filename>

Ex) ./mkbootimg --second boot.img-second.gz --kernel boot.img-kernel --ramdisk boot.img-ramdisk.gz --cmdline "init=/init pci=noearly console=logk0 console=ttyS0 earlyprintk=pti,keep panic\_on\_bad\_page=1 panic\_on\_list\_corruption=1 loglevel=4 vmalloc=256M androidboot.hardware=mofd\_v1 androidboot.spid="xxxx:xxxx:xxxx:xxxx:xxxx:xxxx" androidboot.serialno=01234567890123456789 androidboot.selinux=permissive snd\_pcm.maximum\_substreams=8 ptrace.ptrace\_can\_access=1 allow\_factory=1 ip=50.0.0.2:50.0.0.1::255.255.255.0::usb0:on debug\_locks=0 n\_gsm.mux\_base\_conf=\"ttyACM0,0 ttyXMM0,1\" bootboost=1" --output boot-new.img

The new target\_boot.img can be pushed into a device

adb reboot bootloader

fastboot flash boot target\_boot-new.img

fastboot continue