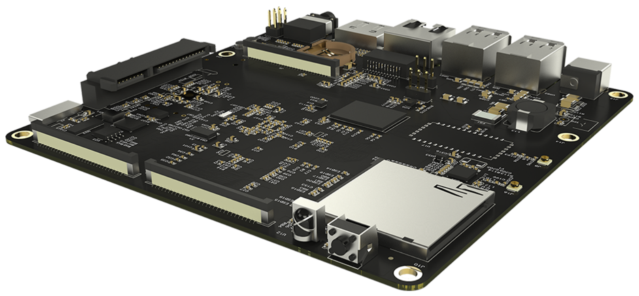
LOFT-Q Quick Guide

Organization: Mixtile Team

Author: Phil.Han <phil.han@focalcrest.com>

Version: 0.1

Date: 2015.01.23



LOFT-Q is the second prototyping board of Mixtile project. Based on AllWinner A31 SOC, designed for embedded developers, engineers, makers and hackers, it can be used as home media center, personal could device, NAS, etc. This guide will help developers quickly building the environment, compiling usable os and customizing you own application.

Build Environment

The board has burned with the latest stable os when you get it. And if you want to customize the board add your own external devices, you will have to rebuild your own system image, then you will have to make some preparation as below.

Source Architecture

The main source of LOFT-Q as below:

- documents: https://github.com/mixtile/loftq-docs
- build: https://github.com/mixtile/loftq-build
- uboot: https://github.com/mixtile/loftq-uboot
- linux: https://github.com/mixtile/loftq-linux
- buildroot: https://github.com/mixtile/buildroot
- android: https://bitbucket.org/Mixtile/loftq-android

Building Environment

Now we can follow steps as below, download different parts of the code, and build the developing environment.

- Create building folder

Follow the following instructions to make root folder.

```
mkdir loftq
cd loftq
```

- Download code repositories

  As previous description, we will have to download the kernel, buildroot, uboot, etc. it will need some time to download all the code repositories.

```
git clone https://github.com/mixtile/loftq-build.git
git clone https://github.com/mixtile/loftq-uboot.git
git clone https://github.com/mixtile/loftq-linux.git
git clone https://github.com/mixtile/buildroot.git
```

About loftq-build

loftq-build contains the srcipts and tools for building uboot, linux, android, and packing system image.

Before accurately building, we will have to import the building environments to current working command line.

```
source loftq-build/sunxi_env.sh
```

After environment importing, we can start compiling instructions, such as building uboot for linux:

```
linux_build_uboot
```

About sunxi_env

sunxi_env.sh is the env importing script for LOFT-Q, working as the lunch script for Android. it will import environment variables and compiling instructions to current working shell.

At the front, it's the definations of the repository paths for uboot, linux, buildroot and android, developers could make some moditfitations according to their own configurations. The following the the original configs:

```
export BUILD_TRUNK=$(pwd)
export BUILD_TRUNK_OUT=$BUILD_TRUNK/out

# envs for sunxi tools
export SUNXI_TOOLS_PATH=$(pwd)/loftq-build
export SUNXI_LINUX_PATH=$(pwd)/loftq-linux
export SUNXI_UBOOT_PATH=$(pwd)/loftq-uboot
export SUNXI_TOOLCHAIN_PATH=${SUNXI_TOOLS_PATH}/toolschain/gcc-linaro/bin/

# envs for android
export ANDROID_TRUNK=$(pwd)/android
export ANDROID_DEVICE=loftq
export ANDROID_DEVICE_TRUNK=${ANDROID_TRUNK}/device/mixtile/${ANDROID_DEVICE}
```

```
# envs for ubuntu touch
# only used if we have android base sdk released by ubuntu touch team
# Note: now we can build this image but can't burn it to disk with PhoenixTool
export UBUNTU_OUTPUT=$BUILD_TRUNK_OUT/ubuntu
export UBUNTU_TARBALL=$UBUNTU_OUTPUT/vivid-preinstalled-touch-armhf.tar.gz

# commom env
export ANDROID_OUT=${ANDROID_TRUNK}/out
export ANDROID_DEVICE_OUT=${ANDROID_OUT}/target/product/${ANDROID_DEVICE}

export LINARO_GCC_PATH=$SUNXI_TOOLCHAIN_PATH
export PATH=$PATH:$LINARO_GCC_PATH
```

And developers can also add their own compiling instructions, sample as *linux_build_uboot* :

```
function linux_build_uboot()
{
    CURDIR=$PWD

    cd $SUNXI_UBOOT_PATH
    make distclean
    make sun6i_config
    make -j4
    cd $CURDIR
}
```

Uboot Building

As for building of uboot, we have two methods. we can build it with predefined building instruction in sunxi_env and follow the manual commands step by step.

Predefined building

we have predefined building instructions uboot for android and linux.

- Uboot for Linux

  ```
  linux_build_uboot
  ```

- Uboot for Android

  ```
  android_build_uboot
  ```

Manually building

Manually building is just the seperate instructions of predefined instruction. Commands as below:

```
make distclean
make sun6i_config
make -j4
```

Linux Kernel Building

Now the kernel we use is based on the cuszomized version by Allwinner for A31 soc, whihc is linux 3.3 . this version contains the drivers and configurations for both android and common linux releases, and we can compile kernel for both GNU/Linux and Android.

Predefined building

loftq-build also provides predefined instructions for kernel building.

- kernel for GNU/Linux

```
linux_build_kernel
```

- kernel for Android

```
android_build_kernel
```

Manually building

Manually building for kernel will be a little complex, which needs external toolchain support.

- toolchain importing

```
export SUNXI_TOOLS_PATH=$(pwd)/loftq-build
export SUNXI_TOOLCHAIN_PATH=$SUNXI_TOOLS_PATH/toolschain/gcc-linaro/bin/
export PATH=$PATH:$SUNXI_TOOLCHAIN_PATH
```

  Developers can add these variables according to own path definations.

- kernel building for GNU/Linux

```
cd loftq-linux
make distclean
./build.sh -p sun6i
```

- kernel building for Android

```
cd loftq-linux
make distclean
./build.sh -p sun6i_fiber
```

Buildroot building

Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation.

Getting source

Now we have the latest buildroot support, developers can download the code from our github repository, which will bee updated with the official repository.

```
git clone https://github.com/mixtile/buildroot.git
```

Compiling buildroot

After downloading source, we can begin compiling buildroot.

```
make mixtile_loftq_defconfig
make
```

Note

- When building with make, it will try to downlaod all the code of package configured, also some temp files will be generated, so before building, it's necessary to prepare enough disk space according to number of packages, or it may cause building failed.
- after building finished, there will be **output/images** at the root building directory.

Packing image

This step is packing image, we will have to put *rootfs.ext4* generated in previous step into the specified directory **$BUILD_TRUNK/out/linux** and then run cmds:

```
linux_pack
```

As for linux image packing, it will show one prompt session as below:

```
nano@vps:~/mixtile/loftq$ linux_pack
Generating linux out directory!
Copiing uboot!
Copying linux kernel and modules!
Packing final image!
Start packing for Lichee system

All valid chips:
0. sun6i
Please select a chip:
```

then enter **0** for choosing **sun6i**, the next session for os :

```
All valid platforms:
0. android
1. dragonboard
2. linux
Please select a platform:
```

Now, we use linux, so enter **2** for **linux**, and then:

```
All valid boards:
0. aw_w01
1. aw_w02
2. evb
3. loftq
4. loftq_suse
5. qc
Please select a board:
```

we use **loftq**, so enter **3**, maybe the options list will vary with time. so choose the specified option according to your request. and then it will continue packing according to your selections, screen info as below:

```
....

c:\bat
c:\magic.bin
find magic !!
RealLen=0x5A5C00
CPlugin Free lib
CPlugin Free lib
get rootfs from ../../../out/linux
compute signature for datafile /home/nano/mixtile/loftq/loftq-build/pack/out/boot.fex
/home/nano/mixtile/loftq/loftq-build/pack/pctools/linux/eDragonEx/
/home/nano/mixtile/loftq/loftq-build/pack/out
Begin Parse sys_partion.fex
Add partion bootloader.fex BOOTLOADER_FEX00
Add partion very bootloader.fex BOOTLOADER_FEX00
FilePath: bootloader.fex
FileLength=5a5c00 FileSizeHigh=0
Add partion env.fex ENV_FEX000000000
Add partion very env.fex ENV_FEX000000000
FilePath: env.fex
FileLength=20000 FileSizeHigh=0
Add partion boot.fex BOOT_FEX00000000
Add partion very boot.fex BOOT_FEX00000000
FilePath: boot.fex
FileLength=dac800 FileSizeHigh=0
Add partion rootfs.fex ROOTFS_FEX000000
Add partion very rootfs.fex ROOTFS_FEX000000
FilePath: rootfs.fex
FileLength=8327800 FileSizeHigh=0
BuildImg 0
Dragon execute image.cfg SUCCESS !
---------image is at-------------

 /home/nano/mixtile/loftq/loftq-build/pack/sun6i_linux_loftq.img

pack finish
```

/home/nano/mixtile/loftq/loftq-build/pack/sun6i_linux_loftq.img is the target image that we need. And next, we can burn this image to sdcard for installing or booting with **PhoenixCard**.

Note

PhoenixCard is the tool provided by Allwinner for burning sdcard for booting or factory installing, which is only for windows platform. so we can't fully be free, and with mainline kernel or uboot, we can fly without this tool ;) , but for now, we still need this.

Customizing

We have preadded packages in configuration file, also developers can add or delete them according to requirement. Commands as below:

```
make menuconfig
```

menuconfig needs some extra libs or commands support, it you have been warned something missed, you can install them according to the prompting infomation.

More Info

Sites of Buildroot:

- Official site: http://buildroot.uclibc.org
- Documents: http://buildroot.uclibc.org/docs.html

Ubuntu building

ubuntu support is still under working, please wait ...

Android building

The android source is based on **Android 4.4.2** provided by Allwinner with drivers for bluetooth, wifi, spdif, inner disk and more. The source tarball needs really much space that we can't afford it with one single code repository, so later, we will release latest code archive in our site. And for now we have one repository for Android code, because of its big body, only readable for downloading.

Building steps as below:

- enter android directory.

  ```
  cd android
  ```

- build uboot for android with predefined instruction.

  ```
  android_build_uboot
  ```

- build kernel for android with predefined instruction.

  ```
  android_build_kernel
  ```

- build android project.

```
source build/envsetup.sh
lunch mars_loftq-eng
android_extract_bsp
make -j16
android_pack
```

OpenSUSE

openSUSE is one of the best GNU/Linux distribution in the world, which is from German. Now we can boot up and try openSUSE JeOS for ARM with customized Uboot and Linux kernel.

JeOS rootfs

There are several versions of JeOS that can work on LOFT-Q.

- Factory: http://download.opensuse.org/ports/armv7hl/factory/images/
- 13.1: http://download.opensuse.org/ports/armv7hl/distribution/13.1/appliances/
- 12.3: http://download.opensuse.org/ports/armv7hl/distribution/12.3/images/

We can choose one version for testing. More info about different versions, please refer to https://en.opensuse.org/HCL:Chroot .

Generating image

Here we take 13.1 version as testing example. then we have to download the **openSUSE-*-ARM-JeOS.armv7-rootfs-*.tbz** tarball. then we follow these steps to generate **rootfs.ext4** :

1. build uboot and linux kernel for GNU/Linux.
2. uncompress JeOS tarball to openSUSE-JeOS directory.
3. generate rootfs.ext4.

   ```
   ./loftq-build/rootfs2ext4.sh -d ./openSUSE-JeOS -t ./rootfs.ext4
   ```

4. copy rootfs.ext4 to **$BUILD_TRUNK/out/linux**.

   ```
   cp rootfs.ext4 ./out/linux/
   ```

5. generate image for PhoenixCard.

   ```
   linux_pack
   ```

6. burn target image generated in last step to sdcard with **PhoenixCard**.

Now, we have one bootable sd card for testing openSUSE, have fun ;).