U-boot

U-boot must be cross-compiled on an x86 Linux PC.

Toolchain

Click one of the site to download toolchain to build U-boot.

- Download #1
- Download #2

Once the download is done, extract the tarball to **/opt/toolchains/**.

```
$ sudo mkdir -p /opt/toolchains
$ sudo tar xvf gcc-linaro-aarch64-none-elf-4.9-2014.09_linux.tar.xz -C
/opt/toolchains/
```

In order to add the toolchain path to PATH, paste below lines to **$HOME/.bashrc**.

```
export ARCH=arm
export CROSS_COMPILE=aarch64-none-elf-
export PATH=/opt/toolchains/gcc-linaro-aarch64-none-
elf-4.9-2014.09_linux/bin/:$PATH
```

You can apply the change if you login again or import to apply this change, login again or evaluate **$HOME/.bashrc** with source command.

```
$ source ~/.bashrc
```

You can check if the toolchain installed above works properly while checking the version of toolchain. If you can find **gcc version 4.9.2 20140904 (prerelease)** at the end of the line, the toolchain is well installed.

```
$ aarch64-none-elf-gcc -v
Using built-in specs.
COLLECT_GCC=aarch64-none-elf-gcc
COLLECT_LTO_WRAPPER=/opt/toolchains/gcc-linaro-aarch64-none-
elf-4.9-2014.09_linux/bin/../libexec/gcc/aarch64-none-elf/4.9.2/lto-wrapper
Target: aarch64-none-elf
Configured with: /cbuild/slaves/oorts/crosstool-ng/builds/aarch64-none-elf-
linux/.build/src/gcc-linaro-4.9-2014.09/configure --build=i686-build_pc-
linux-gnu --host=i686-build_pc-linux-gnu --target=aarch64-none-elf --
prefix=/cbuild/slaves/oorts/crosstool-ng/builds/aarch64-none-elf-
linux/install --with-local-prefix=/cbuild/slaves/oorts/crosstool-
ng/builds/aarch64-none-elf-linux/install/aarch64-none-elf --without-headers
--with-newlib --enable-threads=no --disable-shared --with-
pkgversion='crosstool-NG linaro-1.13.1-4.9-2014.09 - Linaro GCC 4.9-2014.09'
```

```
--with-bugurl=https://bugs.launchpad.net/gcc-linaro --with-arch=armv8-a --
disable-__cxa_atexit --with-gmp=/cbuild/slaves/oorts/crosstool-
ng/builds/aarch64-none-elf-linux/.build/aarch64-none-elf/build/static --
with-mpfr=/cbuild/slaves/oorts/crosstool-ng/builds/aarch64-none-elf-
linux/.build/aarch64-none-elf/build/static --with-
mpc=/cbuild/slaves/oorts/crosstool-ng/builds/aarch64-none-elf-
linux/.build/aarch64-none-elf/build/static --with-
isl=/cbuild/slaves/oorts/crosstool-ng/builds/aarch64-none-elf-
linux/.build/aarch64-none-elf/build/static --with-
cloog=/cbuild/slaves/oorts/crosstool-ng/builds/aarch64-none-elf-
linux/.build/aarch64-none-elf/build/static --with-
libelf=/cbuild/slaves/oorts/crosstool-ng/builds/aarch64-none-elf-
linux/.build/aarch64-none-elf/build/static --enable-lto --enable-linker-
build-id --disable-libmudflap --disable-libgomp --disable-libssp --disable-
libstdcxx-pch --enable-multiarch --disable-multilib --enable-
languages=c,c++,fortran
Thread model: single
gcc version 4.9.2 20140904 (prerelease) (crosstool-NG
linaro-1.13.1-4.9-2014.09 - Linaro GCC 4.9-2014.09)
```

Checkout & compile

You can checkout **U-boot** source tree from **Hardkernel's Github**.

```
$ git clone https://github.com/hardkernel/u-boot.git -b odroidc2-v2015.01
```

Before you compile **U-boot**, you must configure for **ODROID-C2** with following command.

```
$ cd u-boot
$ make odroidc2_config
```

And finally you can compile **U-boot**.

```
$ make
```

Installation

The bootloader for **ODROID-C2** is consisted with two part, **1st stage bootloader** + **U-boot**. You can find **u-boot.bin** once you compile **U-boot**, as well as **bl1.bin.hardkernel** in the directory **sd_fuse/**. Both binaries must be installed in a card (eMMC or MicroSD).

- The 1st stage bootloader, **bl1.bin.hardkernel**, is provided as prebuilt binary only.
- The u-boot binary needs to be signed. This is done by an x86-64 executable, **aml_encrypt_gxb**, which is shipped with the u-boot source; source for aml_encrypt_gxb is not available..

Installation to blank card

We provide the script, **sd_fuse/sd_fusing.sh**, this helps you to install the bootloader into your blank card eMMC or MicroSD.

1. Insert your card to USB card reader and attach to USB host port of your desktop.
2. Check the device path of your USB card reader.
3. Install the bootloader binaries using **sd_fuse/sd_fusing.sh**.

```
$ cd sd_fuse
$ ./sd_fusing.sh <device/path/of/your/card>
```

You can download and install the latest version of binaries as following.

Click the following site to download the boot loader of final release version

- Download - Linux Ubuntu/Android (update Dec 22, 2017)

```
$ tar zxvf c2_boot_release.tar.gz
$ cd sd_fuse
$ ./sd_fusing.sh <device/path/of/your/card>
```

Installation using fastboot

If you can boot your **ODROID-C2** already and want to install a new **u-boot.bin** built by you. Fastboot helps you to install a **u-boot.bin** into your board.

```
$ sudo fastboot flash bootloader sd_fuse/u-boot.bin
```

If installation is done, you care reboot your **ODROID-C2** with fastboot.

```
$ sudo fastboot reboot
```

Installation from Linux on C2

You can also install U-boot from Linux with **dd** command. After building U-boot, copy **u-boot.bin** into your **ODROID-C2**. Then do the command below in order to flash your U-boot image to MicroSD or eMMC.

```
$ sudo dd if=u-boot.bin of=/dev/mmcblk0 bs=512 seek=97
$ sudo sync
```

- You must use the u-boot.bin binary under the folder, “u-boot/sd_fuse/u-boot.bin”

Update from Linux on ODROID-C2 board using u-boot package

On ODROID-C2 ubuntu, you can update U-boot with the latest released one using apt-get install command.

```
$ sudo apt-get install u-boot
```

TFTP

HOST-PC

- **NOTE:** *This step is Ubuntu 14.04 example.*

Install the package:

```
$ sudo apt-get install tftpd-hpa
```

Edit config file as follows:

```
$ sudo vi /etc/default/tftpd-hpa
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="-s -c -l"
```

Create TFTP folder:

```
sudo mkdir /tftpboot
sudo chmod -R 777 /tftpboot
sudo chown -R nobody /tftpboot
```

Restart the app to apply new configuration:

```
sudo service tftpd-hpa restart
```

Copy *Image*, *meson64_odroidc2.dtb* and *uInitrd* files to */tftpboot* directory.

U-Boot

Assumptions:

- TFTP server IP address: *192.168.0.2*
- ODROID-C2 IP address: *192.168.0.3*

Set the tftp server & board ip address:

```
odroidc2# setenv serverip 192.168.0.2
odroidc2# setenv ipaddr 192.168.0.3
```

Download images using tftp:

```
odroidc2# tftp 0x1000000 meson64_odroidc2.dtb
odroidc2# tftp 0x11000000 Image
odroidc2# tftp 0x13000000 uInitrd
```

Booting:

```
odroidc2# setenv condev "console=ttyS0,115200n8 console=tty0"
odroidc2# setenv m "1080p60hz"
odroidc2# setenv m_bpp "32"
odroidc2# setenv bootargs "root=UUID=e139ce78-9841-40fe-8823-96a304a09859
rootwait rw ${condev} no_console_suspend hdmimode=${m} m_bpp=${m_bpp}
fsck.fix=yes"
odroidc2# booti 0x11000000 0x13000000 0x1000000
```

How to build u-boot with bl301 firmware

Because of the issue of build failure on any platform other than x86-64 and support of buildroot, we've separated bl301 firmware source code from u-boot repository.

In the path, u-boot/fip/gxb/, there is a pre-built bl301.bin binary of the latest version, so you can build just u-boot without a re-build bl301 process.

If you want to modify and update bl301-related source codes by yourself, please refer to the following.

Checkout and Merge

You can checkout **bl301 firmware** source tree from **Hardkernel's Github**.

```
$ git clone https://github.com/hardkernel/u-boot_firmware.git -b odroidc2-
bl301
```

You should merge bl301 with u-boot before build it.

```
$ git clone https://github.com/hardkernel/u-boot.git -b odroidc2-v2015.01
$ git clone https://github.com/hardkernel/u-boot_firmware.git -b odroidc2-
bl301

$ cd u-boot

$ git remote add u-boot_firmware ../u-boot_firmware/
$ git fetch u-boot_firmware
```

```
$ git merge --no-commit u-boot_firmware/odroidc2-bl301
```

Toolchain for bl301

bl301.bin source code needs a toolchain, arm-none-eabi-. Click one of the site to download toolchain to build U-boot.

- Download

Once the download is done, extract the tarball to **/opt/toolchains/**.

```
$ sudo mkdir -p /opt/toolchains
$ sudo tar xvf gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz -C
/opt/toolchains/
```

In order to add the toolchain path to PATH, paste below lines to **$HOME/.bashrc**.

```
export PATH=/opt/toolchains/gcc-linaro-arm-none-
eabi-4.8-2014.04_linux/bin/:$PATH
```

You can apply the change if you login again or import to apply this change, login again or evaluate **$HOME/.bashrc** with source command.

```
$ source ~/.bashrc
```

You can check if the toolchain installed above works properly while checking the version of toolchain.

```
Using built-in specs.
COLLECT_GCC=arm-none-eabi-gcc
COLLECT_LTO_WRAPPER=/opt/toolchains/gcc-linaro-arm-none-
eabi-4.8-2014.04_linux/bin/../libexec/gcc/arm-none-eabi/4.8.3/lto-wrapper
Target: arm-none-eabi
Configured with: /cbuild/slaves/oorts/crosstool-ng/builds/arm-none-eabi-
linux/.build/src/gcc-linaro-4.8-2014.04/configure --build=i686-build_pc-
linux-gnu --host=i686-build_pc-linux-gnu --target=arm-none-eabi --
prefix=/cbuild/slaves/oorts/crosstool-ng/builds/arm-none-eabi-linux/install
--with-local-prefix=/cbuild/slaves/oorts/crosstool-ng/builds/arm-none-eabi-
linux/install/arm-none-eabi --without-headers --with-newlib --enable-
threads=no --disable-shared --with-pkgversion='crosstool-NG
linaro-1.13.1-4.8-2014.04 - Linaro GCC 4.8-2014.04' --with-
bugurl=https://bugs.launchpad.net/gcc-linaro --disable-__cxa_atexit --with-
gmp=/cbuild/slaves/oorts/crosstool-ng/builds/arm-none-eabi-linux/.build/arm-
none-eabi/build/static --with-mpfr=/cbuild/slaves/oorts/crosstool-
ng/builds/arm-none-eabi-linux/.build/arm-none-eabi/build/static --with-
mpc=/cbuild/slaves/oorts/crosstool-ng/builds/arm-none-eabi-linux/.build/arm-
none-eabi/build/static --with-isl=/cbuild/slaves/oorts/crosstool-
ng/builds/arm-none-eabi-linux/.build/arm-none-eabi/build/static --with-
cloog=/cbuild/slaves/oorts/crosstool-ng/builds/arm-none-eabi-
linux/.build/arm-none-eabi/build/static --with-
```

```
libelf=/cbuild/slaves/oorts/crosstool-ng/builds/arm-none-eabi-
linux/.build/arm-none-eabi/build/static --enable-lto --enable-linker-build-
id --enable-libmudflap --disable-libgomp --enable-libssp --disable-
libstdcxx-pch --enable-multilib --enable-languages=c,c++,fortran --with-
multilib-list=aprofile
Thread model: single
gcc version 4.8.3 20140401 (prerelease) (crosstool-NG
linaro-1.13.1-4.8-2014.04 - Linaro GCC 4.8-2014.04)
```

Build

And finally you can compile u-boot with bl301 firmware.

```
$ cd u-boot
$ make distclean
$ make odroidc2_config
$ make
```

How to check U-Boot version on User Space

You can check the U-Boot version using **dd** command as following.

Ubunut

```
root@odroid64:/home/odroid# vi getUbootVer.sh

#!/bin/sh
dd if=/dev/mmcblk0 of=/tmp/temp.bin bs=512 skip=97 count=1334 status=none
grep -a -r -E -o ".{0,0}U-Boot 2015.01.{0,50}" /tmp/temp.bin | grep -a "("
rm /tmp/temp.bin

root@odroid64:/home/odroid# chmod a+x getUbootVer.sh

root@odroid64:/home/odroid# ./getUbootVer.sh
    U-Boot 2015.01-00167-g095fdbe-dirty (Nov 12 2016 - 16:35:32)
```

Android

```
shell@odroidc2:/ $ su
root@odroidc2:/ # mount -o rw,remount /
root@odroidc2:/ # vi getUbootVer.sh

#!/bin/sh
dd if=/dev/block/mmcblk0 of=/temp.bin bs=512 skip=97 count=1334
grep -a -r -E -o ".{0,0}U-Boot 2015.01.{0,50}" /temp.bin | grep -a "("
```

```
rm /temp.bin

root@odroidc2:/ # chmod 755 /getUbootVer.sh

root@odroidc2:/ # /getUbootVer.sh
1334+ records in
1334+ records out
683008 bytes transferred in 0.014 secs (48786285 bytes/sec)
/temp.bin:U-Boot 2015.01-dirty (Jul 06 2017 - 07:47:27)
```