This is a page about TI's Cortex-A8 based; BeagleBone Black.

Availability

Boards:

BeagleBone Black at Digi-Key

BeagleBone Green at Digi-Key

Embest BeagleBone Black at Digi-Key

BeagleBone Black Wireless at Digi-Key

BeagleBone Green Wireless at Digi-Key

Power Supplies:

USB Micro for BeagleBone Green at Digi-Key

Cables:

(USB to serial adapter) TTL-232R-3V3 at Digi-Key

HDMI-A Male to HDMI-D Male (1.5M) at Digi-Key

HDMI-A Male to HDMI-D Male (1.5M) at Digi-Key

HDMI-A Male to HDMI-D Male (2M) at Digi-Key

Basic Requirements

- Running a recent release of Debian, Fedora or Ubuntu; without OS Virtualization Software.
- ARM Cross Compiler – Linaro: http://www.linaro.org
  - Linaro Toolchain Binaries: http://www.linaro.org/downloads/
- Bootloader
  - Das U-Boot – the Universal Boot Loader: http://www.denx.de/wiki/U-Boot
  - Source: http://git.denx.de/?p=u-boot.git;a=summary
- Linux Kernel
  - Linus's Mainline tree: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git
- ARM based rootfs
  - Debian: https://www.debian.org
  - Ubuntu: http://www.ubuntu.com

ARM Cross Compiler: GCC

This is a pre-built (64bit) version of Linaro GCC that runs on generic linux, sorry (32bit) x86 users, it's time to upgrade...

Download/Extract:

~/

```
wget -c https://releases.linaro.org/components/toolchain/binaries/6.4-2018.05/arr
tar xf gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
export CC=`pwd`/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf/bin/arm-linu
```

Test Cross Compiler:

~/

```
${CC}gcc --version
arm-linux-gnueabihf-gcc (Linaro GCC 6.4-2018.05) 6.4.1 20180425 [linaro-6.4-2018
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

注意：可以将交叉编译工具链加入到环境变量中

sudo vim /etc/profile

#add for arm-linux- cross compiler by hero
export PATH="${PATH}:/home/hero/BBB/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf/bin/"

注意：/etc/profile文件操作不当会导致无法登陆，请严格按照上面的操作

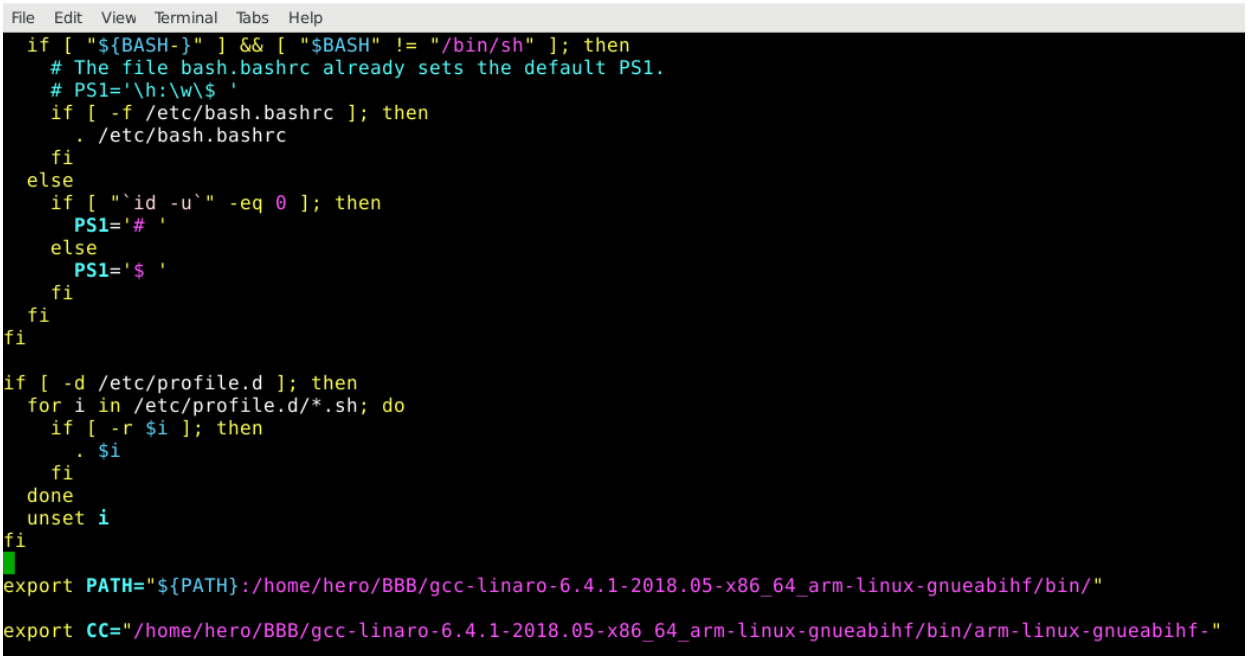
```
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
export PATH

if [ "${PS1-}" ]; then
  if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

#add for arm linux  cross compiler by hero
export PATH="${PATH}:/home/hero/BBB/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf/bin/"
```

或者在~/.bashrc文件末尾添加



标准化安装交叉编译器

我们在使用交叉编译器的时候一般是采用arm-tonghuix-linux-gnueabi这样的命令的，但是很多标准Makefile需要实用标准的交叉编译器的名称，一般这个名称是arm-linux-gcc这样的。

`cd /opt/arm-tonghuix-linux-gnueabi/bin/`

那么我们在生成交叉编译器的目录下写一个link.sh脚本，新建一些软链接。

>link.sh

#!/bin/sh

PREFIX=arm-linux-gnueabihf-

AFTFIX=arm-linux-

ln -s ${PREFIX}addr2line ${AFTFIX}addr2line

ln -s ${PREFIX}ar ${AFTFIX}ar

ln -s ${PREFIX}as ${AFTFIX}as

ln -s ${PREFIX}c++ ${AFTFIX}c++

ln -s ${PREFIX}c++filt ${AFTFIX}c++filt

ln -s ${PREFIX}cpp ${AFTFIX}cpp

ln -s ${PREFIX}dwp ${AFTFIX}dwp

ln -s ${PREFIX}elfedit ${AFTFIX}elfedit

ln -s ${PREFIX}g++ ${AFTFIX}g++

ln -s ${PREFIX}gcc ${AFTFIX}gcc

ln -s ${PREFIX}gcc-6.4.1 ${AFTFIX}gcc-6.4.1

ln -s ${PREFIX}gcc-ar ${AFTFIX}gcc-ar

ln -s ${PREFIX}gcc-nm ${AFTFIX}gcc-nm

ln -s ${PREFIX}gcc-ranlib ${AFTFIX}gcc-ranlib

ln -s ${PREFIX}gcov ${AFTFIX}gcov
ln -s ${PREFIX}gcov-dump ${AFTFIX}gcov-dump
ln -s ${PREFIX}gcov-tool ${AFTFIX}gcov-tool
ln -s ${PREFIX}gdb ${AFTFIX}gdb
ln -s ${PREFIX}gfortran ${AFTFIX}gfortran
ln -s ${PREFIX}gprof ${AFTFIX}gprof
ln -s ${PREFIX}ld ${AFTFIX}ld
ln -s ${PREFIX}ld.bfd ${AFTFIX}ld.bfd
ln -s ${PREFIX}ld.gold ${AFTFIX}ld.gold
ln -s ${PREFIX}nm ${AFTFIX}nm
ln -s ${PREFIX}objcopy ${AFTFIX}objcopy
ln -s ${PREFIX}objdump ${AFTFIX}objdump
ln -s ${PREFIX}ranlib ${AFTFIX}ranlib
ln -s ${PREFIX}readelf ${AFTFIX}readelf
ln -s ${PREFIX}size ${AFTFIX}size
ln -s ${PREFIX}strings ${AFTFIX}strings
ln -s ${PREFIX}strip ${AFTFIX}strip

写完这个link.sh文件以后，我们可以执行一下

```
sh link.sh
```

这样我们就得到了整个使用标准名称的交叉编译工具链，可以在makefile中使用类似arm-linux-gcc这样的名称了。

Bootloader: U-Boot

Das U-Boot – the Universal Boot Loader: http://www.denx.de/wiki/U-Boot

eewiki.net patch archive: https://github.com/eewiki/u-boot-patches

建议将克隆的库放在一个空文件夹里（我们使用common这个目录），然后将BeagleBone Black的文件存放在本地的另一个目录中（我们使用BBB这个目录）。

以下是原参考，

Download:

~/

```
git clone https://github.com/u-boot/u-boot
cd u-boot/
git checkout v2018.09 -b tmp
```

我们改为这样执行：

```
mkdir ~/common
cd ~/common
git clone --bare https://github.com/u-boot/u-boot

cd ../BBB/
git clone --reference ~/common/u-boot.git
https://github.com/u-boot/u-boot

cd u-boot/
```

git checkout v2018.09 -b tmp

Patches:

~/u-boot

```
wget -c https://rcn-ee.com/repos/git/u-boot-patches/v2018.09/0001-am335x_evm-uEnv
wget -c https://rcn-ee.com/repos/git/u-boot-patches/v2018.09/0002-U-Boot-BeagleBo

patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
patch -p1 < 0002-U-Boot-BeagleBone-Cape-Manager.patch
```

wget -c https://rcn-ee.com/repos/git/u-boot-patches/v2018.09/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch

wget -c https://rcn-ee.com/repos/git/u-boot-patches/v2018.09/0002-U-Boot-BeagleBone-Cape-Manager.patch

Configure and Build:

~/u-boot

```
make ARCH=arm CROSS_COMPILE=${CC} distclean
make ARCH=arm CROSS_COMPILE=${CC} am335x_evm_defconfig
make ARCH=arm CROSS_COMPILE=${CC}
```

Linux Kernel

Spectre: https://meltdownattack.com and https://developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability

Spectre v2 Migation requires minimal: U-Boot: v2018.07 and Kernel: v4.18.x

```
[    0.047363] CPU0: Spectre v2: using BPIALL workaround
```

This script will build the kernel, modules, device tree binaries and copy them to the deploy directory.

Mainline

Download:

~/

```
git clone https://github.com/RobertCNelson/bb-kernel
cd bb-kernel/
```

For am33x-v4.9 (Longterm 4.9.x):

~/bb-kernel/

```
git checkout origin/am33x-v4.9 -b tmp
```

For am33x-rt-v4.9 (Longterm 4.9.x + Real-Time Linux):

~/bb-kernel/

```
git checkout origin/am33x-rt-v4.9 -b tmp
```

For am33x-v4.14 (Longterm 4.14.x):

~/bb-kernel/

```
git checkout origin/am33x-v4.14 -b tmp
```

For am33x-rt-v4.14 (Longterm 4.14.x + Real-Time Linux):

~/bb-kernel/

```
git checkout origin/am33x-rt-v4.14 -b tmp
```

For am33x-v4.18 (Stable):

~/bb-kernel/

```
git checkout origin/am33x-v4.18 -b tmp
```

同样执行我们在U-Boot中的操作：

因为kernel比较大，所以我们要先做些设置，防止出现以下错误：

```
hero@debian:~/common$ git clone --bare https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
Cloning into bare repository 'linux-stable.git'...
remote: Counting objects: 7334790, done.
remote: Compressing objects: 100% (1097218/1097218), done.
error: RPC failed; curl 56 GnuTLS recv error (-9): A TLS packet with unexpected length was received.
fatal: The remote end hung up unexpectedly
fatal: early EOF
fatal: index-pack failed
```

首先设置buff为1500M

git config --global http.postBuffer 1572864000

git config --global user.name "zyq5428"

git config --global user.email "594270026@qq.com"

然后下载内核：

cd ~/common/

git clone --bare https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git

执行一个小操作，以便于把刚刚下载的数据转换为适合build_kernel.sh这个脚本的形式

mkdir linux-stable

mv linux-stable.git/ linux-stable/.git

cd linux-stable/

git config --local --bool core.bare false

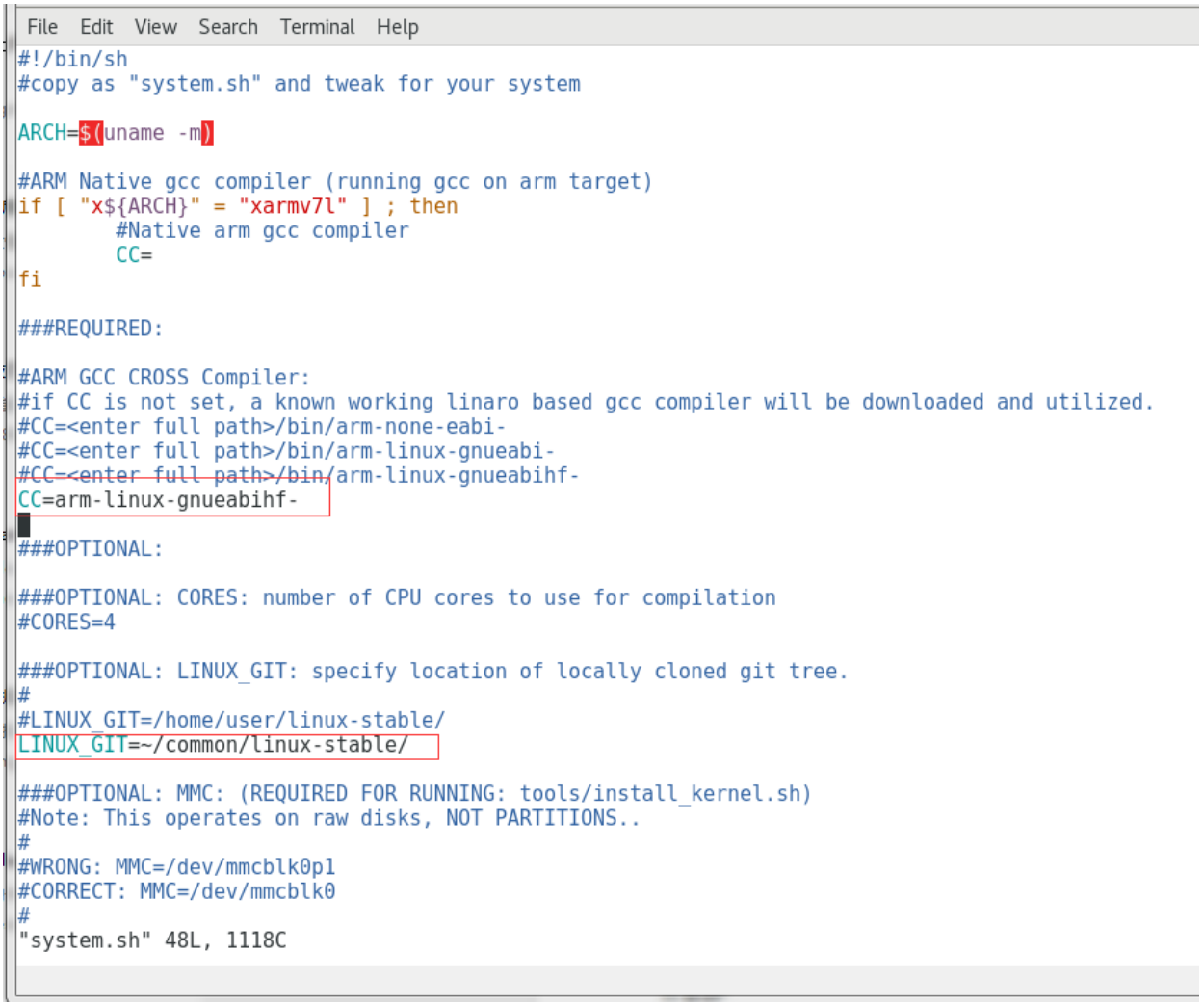
以这种形式，将一个裸库转换成一个正常的库，但没有检查任何文件，并且节省了大量的磁盘空间。

现在，在system.sh文件中正确设置变量

cd bb-kernel/

cp system.sh.sample system.sh

vim system.sh



Build:

~/bb-kernel/

```
./build_kernel.sh
```

TI BSP

Download:

~/

```
git clone https://github.com/RobertCNelson/ti-linux-kernel-dev.git
cd ti-linux-kernel-dev/
```

For TI v4.14.x:

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-4.14.y -b tmp
```

For TI v4.14.x: Real-Time

~/ti-linux-kernel-dev/

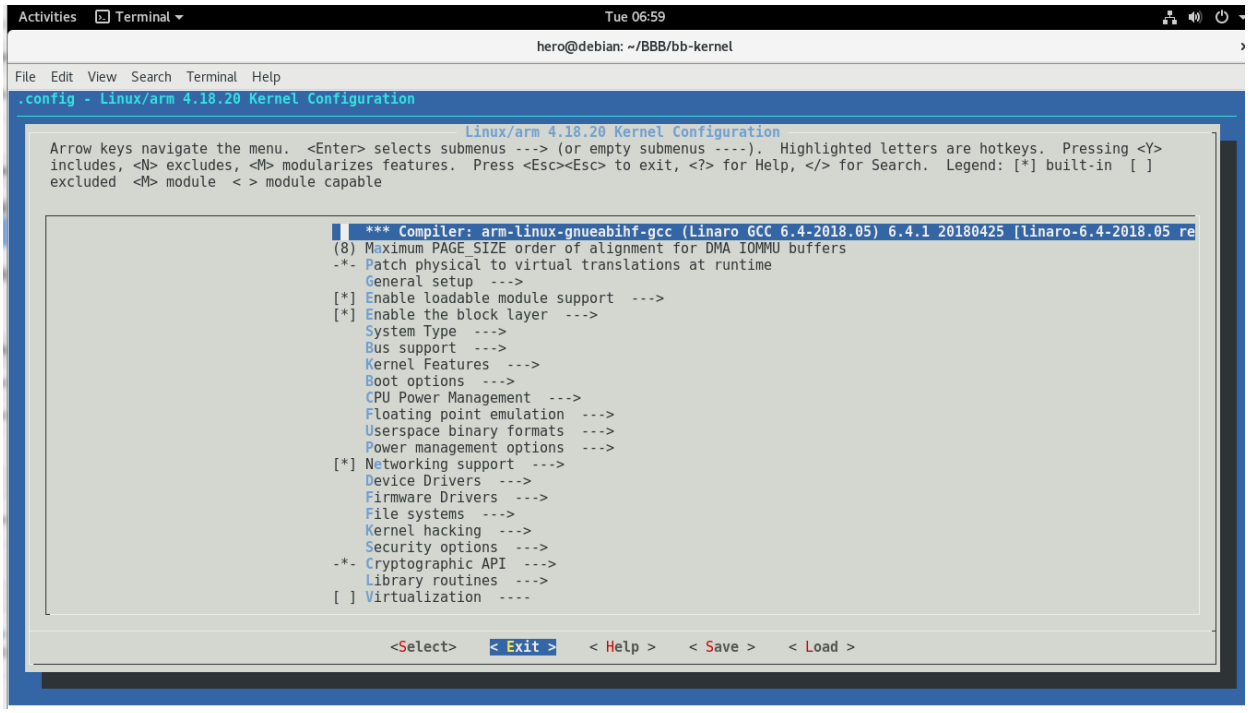
```
git checkout origin/ti-linux-rt-4.14.y -b tmp
```

Build:

~/ti-linux-kernel-dev/

```
./build_kernel.sh
```

到下图时选择Exit菜单选项来确认默认设置，然后继续，




```
-----------------------------
'arch/arm/boot/zImage' -> '/home/hero/BBB/bb-kernel/deploy/4.18.20-bone16.zImage'
'.config' -> '/home/hero/BBB/bb-kernel/deploy/config-4.18.20-bone16'
-rwxr-xr-x 1 hero hero 8.1M Feb 26 09:01 /home/hero/BBB/bb-kernel/deploy/4.18.20-bone16.zImage
-----------------------------
Building modules archive...
Compressing 4.18.20-bone16-modules.tar.gz...
-rw-r--r-- 1 hero hero 17M Feb 26 09:02 /home/hero/BBB/bb-kernel/deploy/4.18.20-bone16-modules.tar.gz
-----------------------------
Building dtbs archive...
Compressing 4.18.20-bone16-dtbs.tar.gz...
-rw-r--r-- 1 hero hero 479K Feb 26 09:02 /home/hero/BBB/bb-kernel/deploy/4.18.20-bone16-dtbs.tar.gz
-----------------------------
Script Complete
eewiki.net: [user@localhost:~$ export kernel_version=4.18.20-bone16]
-----------------------------
hero@debian:~/BBB/bb-kernel$
```

```
export kernel_version=4.18.20-bone16
```

Root File System

Debian 9

User	Password
debian	temppwd
root	root

Download:

~/

```
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/debian-9.5-minimal-armhf-2018-07-3
```

Verify:

~/

```
sha256sum debian-9.5-minimal-armhf-2018-07-30.tar.xz
9399d649d1ce9910bbfc745f59dc57ee0e1134f57e8cae01c8cd75a8bd9d1e1e  debian-9.5-min
```

Extract:

~/

```
tar xf debian-9.5-minimal-armhf-2018-07-30.tar.xz
```

Ubuntu 18.04 LTS

User	Password
ubuntu	temppwd

Download:

~/

```
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/ubuntu-18.04.1-minimal-armhf-2018-
```

Verify:

~/

```
sha256sum ubuntu-18.04.1-minimal-armhf-2018-07-30.tar.xz
6b212ee7dd0d5c9c0af49c22cf78b63e6ad20cec641c303232fca9f21a18804c  ubuntu-18.04.1-
```

Extract:

~/

```
tar xf ubuntu-18.04.1-minimal-armhf-2018-07-30.tar.xz
```

Setup microSD card

For these instruction we are assuming, DISK=/dev/mmcblk0, lsblk is very useful for determining the device id.

```
export DISK=/dev/mmcblk0
```

Erase partition table/labels on microSD card:

```
sudo dd if=/dev/zero of=${DISK} bs=1M count=10
```

Install Bootloader:

~/

```
sudo dd if=./u-boot/MLO of=${DISK} count=1 seek=1 bs=128k
sudo dd if=./u-boot/u-boot.img of=${DISK} count=2 seek=1 bs=384k
```

Create Partition Layout:

With util-linux v2.26, sfdisk was rewritten and is now based on libfdisk.

```
sudo sfdisk --version
sfdisk from util-linux 2.27.1
```

sfdisk >= 2.26.x

```
sudo sfdisk ${DISK} <<-__EOF__
4M,,L,*
__EOF__
```

sfdisk <= 2.25.x

```
sudo sfdisk --unit M ${DISK} <<-__EOF__
4,,L,*
__EOF__
```

Format Partition:

With mkfs.ext4 1.43, we need to make sure metadata_csum and 64bit are disabled.

As the version of U-Boot needed for this target CAN NOT correctly handle reading files with these newer ext4 options.

mkfs.ext4 -V

```
sudo mkfs.ext4 -V
mke2fs 1.43-WIP (15-Mar-2016)
        Using EXT2FS Library version 1.43-WIP
```

mkfs.ext4 >= 1.43

```
for: DISK=/dev/mmcblk0
sudo mkfs.ext4 -L rootfs -O ^metadata_csum,^64bit ${DISK}p1

for: DISK=/dev/sdX
sudo mkfs.ext4 -L rootfs -O ^metadata_csum,^64bit ${DISK}1
```

mkfs.ext4 <= 1.42

```
for: DISK=/dev/mmcblk0
sudo mkfs.ext4 -L rootfs ${DISK}p1

for: DISK=/dev/sdX
sudo mkfs.ext4 -L rootfs ${DISK}1
```

Mount Partition:

On most systems these partitions may will be auto-mounted...

```
sudo mkdir -p /media/rootfs/

for: DISK=/dev/mmcblk0
sudo mount ${DISK}p1 /media/rootfs/

for: DISK=/dev/sdX
sudo mount ${DISK}1 /media/rootfs/
```

Backup Bootloader

This version of MLO/u-boot.img will be used on the "eMMC" flasher script on this page.

~/

```
sudo mkdir -p /media/rootfs/opt/backup/uboot/
sudo cp -v ./u-boot/MLO /media/rootfs/opt/backup/uboot/
sudo cp -v ./u-boot/u-boot.img /media/rootfs/opt/backup/uboot/
```

Install Kernel and Root File System

To help new users, since the kernel version can change on a daily basis. The kernel building scripts listed on this page will now give you a hint of what kernel version was built.

```
-----------------------------
Script Complete
eewiki.net: [user@localhost:~$ export kernel_version=4.X.Y-Z]
-----------------------------
```

Copy and paste that "export kernel_version=4.X.Y-Z" exactly as shown in your own build/desktop environment and hit enter to create an environment variable to be used later.

```
export kernel_version=4.X.Y-Z
```

Copy Root File System

~/

```
sudo tar xfvp ./*-*-*-armhf-*/armhf-rootfs-*.tar -C /media/rootfs/
sync
sudo chown root:root /media/rootfs/
sudo chmod 755 /media/rootfs/
```

Set uname_r in /boot/uEnv.txt

~/

```
sudo sh -c "echo 'uname_r=${kernel_version}' >> /media/rootfs/boot/uEnv.txt"
```

Copy Kernel Image

Kernel Image:

~/

```
sudo cp -v ./bb-kernel/deploy/${kernel_version}.zImage /media/rootfs/boot/vmlinu
```

Copy Kernel Device Tree Binaries

~/

```
sudo mkdir -p /media/rootfs/boot/dtbs/${kernel_version}/
sudo tar xfv ./bb-kernel/deploy/${kernel_version}-dtbs.tar.gz -C /media/rootfs/bo
```

Copy Kernel Modules

~/

```
sudo tar xfv ./bb-kernel/deploy/${kernel_version}-modules.tar.gz -C /media/rootfs
```

File Systems Table (/etc/fstab)

```
sudo sh -c "echo '/dev/mmcblk0p1  /  auto  errors=remount-ro  0  1' >> /media/roo
```

Networking

Edit: /etc/network/interfaces

```
sudo nano /media/rootfs/etc/network/interfaces
```

Add:

/etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

Networking: Using a shared SD card with Multiple BeagleBone

To always enable the Ethernet interface as eth0.

Edit: /etc/udev/rules.d/70-persistent-net.rules

```
sudo nano /media/rootfs/etc/udev/rules.d/70-persistent-net.rules
```

Add:

/etc/udev/rules.d/70-persistent-net.rules

```
# BeagleBone: net device ()
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x0", ATTR{type}==
```

Remove microSD/SD card

```
sync
sudo umount /media/rootfs
```

HDMI

This sections assumes you have already installed your favorite xorg based window manager, such as lxde, xfce, kde, gnome, etc... These are packages that need to be installed on top of your selected windows manager and an xorg.conf needed to correctly setup the video interface.

Note: If the cursor doesn\'t show up right away, first hit: ctlr-alt-f1 then: ctrl-alt-f7 after which it \'should\' show up...

Make sure to install, fbdev driver and xrandr utilities:

```
sudo apt update
sudo apt install read-edid xserver-xorg-video-fbdev x11-xserver-utils
```

/etc/X11/xorg.conf

```
Section "Monitor"
        Identifier      "Builtin Default Monitor"
EndSection
Section "Device"
        Identifier      "Builtin Default fbdev Device 0"
        Driver          "fbdev"
EndSection
Section "Screen"
        Identifier      "Builtin Default fbdev Screen 0"
        Device          "Builtin Default fbdev Device 0"
        Monitor         "Builtin Default Monitor"
EndSection
Section "ServerLayout"
        Identifier      "Builtin Default Layout"
        Screen          "Builtin Default fbdev Screen 0"
EndSection
```

xrandr:

```
xrandr
xrandr --output HDMI-0 --mode 1024x768 --rate 60
```

xrandr (over serial/ssh)

```
xrandr -display :0.0 -q
xrandr -display :0.0 --output HDMI-0 --mode 1024x768 --rate 60
```

eMMC

Script to copy your microSD card to eMMC: (this will need these packages installed: initramfs-tools dosfstools rsync)

```
wget https://raw.githubusercontent.com/RobertCNelson/boot-scripts/master/tools/eM
chmod +x bbb-eMMC-flasher-eewiki-ext4.sh
sudo /bin/bash ./bbb-eMMC-flasher-eewiki-ext4.sh
```

U-Boot Overlays

Full Documentation: readme

Any issues:

Any issues run:

```
sudo /opt/scripts/tools/version.sh
```

Enable:

/boot/uEnv.txt

```
enable_uboot_overlays=1
```

To Disable: eMMC:

/boot/uEnv.txt

```
disable_uboot_overlay_emmc=1
```

To Disable: HDMI VIDEO & AUDIO:

/boot/uEnv.txt

```
disable_uboot_overlay_video=1
```

To Disable: HDMI AUDIO:

/boot/uEnv.txt

```
disable_uboot_overlay_audio=1
```

To Disable: WL1835:

/boot/uEnv.txt

```
disable_uboot_overlay_wireless=1
```

To Disable: BB-ADC:

/boot/uEnv.txt

```
disable_uboot_overlay_adc=1
```

U-Boot: override detected capes

/boot/uEnv.txt

```
uboot_overlay_addr0=/lib/firmware/.dtbo
uboot_overlay_addr1=/lib/firmware/.dtbo
uboot_overlay_addr2=/lib/firmware/.dtbo
uboot_overlay_addr3=/lib/firmware/.dtbo
```

U-Boot: disable auto-loading of detected capes

/boot/uEnv.txt

```
```

```
disable_uboot_overlay_addr0=1
disable_uboot_overlay_addr1=1
disable_uboot_overlay_addr2=1
disable_uboot_overlay_addr3=1
```

U-Boot: load 4 more un-detected capes

/boot/uEnv.txt

```
uboot_overlay_addr4=/lib/firmware/.dtbo
uboot_overlay_addr5=/lib/firmware/.dtbo
uboot_overlay_addr6=/lib/firmware/.dtbo
uboot_overlay_addr7=/lib/firmware/.dtbo
```

U-Boot: PRU Options (v4.14.x-ti)

/boot/uEnv.txt

```
uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-14-TI-00A0.dtbo
```

U-Boot: PRU Options

/boot/uEnv.txt

```
uboot_overlay_pru=/lib/firmware/AM335X-PRU-UIO-00A0.dtbo
```

U-Boot: Cape Universal

/boot/uEnv.txt

```
enable_uboot_cape_universal=1
```

Comments

Comments, feedback, and questions can be sent to: eewiki@digikey.com

Please use the Digi-Key's TechForum: TechForum