BeagleBone Black

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

- Availability
- Basic Requirements
- ARM Cross Compiler: GCC
- Bootloader: U-Boot
- Linux Kernel
  - Mainline
  - TI BSP
- Root File System
  - Debian 9
  - Ubuntu 16.04 LTS
- Setup microSD card
  - Backup Bootloader
  - Dealing with old Bootloader in eMMC
- Install Kernel and Root File System
  - Copy Root File System
  - Set uname_r in /boot/uEnv.txt
  - Copy Kernel Image
  - Copy Kernel Device Tree Binaries
  - Copy Kernel Modules
  - File Systems Table (/etc/fstab)
  - Networking
  - Networking: Using a shared SD card with Multiple BeagleBone
  - Remove microSD/SD card
  - HDMI
  - eMMC
  - SGX
  - U-Boot Overlays
- Comments

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/6.4-2017.11/arm-
linux-gnueabihf/gcc-linaro-6.4.1-2017.11-x86_64_arm-linux-gnueabihf.tar.xz
tar xf gcc-linaro-6.4.1-2017.11-x86_64_arm-linux-gnueabihf.tar.xz
export
CC=`pwd`/gcc-linaro-6.4.1-2017.11-x86_64_arm-linux-gnueabihf/bin/arm-linux
-gnueabihf-
```

Test Cross Compiler:

~/

```
${CC}gcc --version
arm-linux-gnueabihf-gcc (Linaro GCC 6.4-2017.11) 6.4.1 20171012
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.
```

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
Download:

~/

```
git clone https://github.com/u-boot/u-boot
cd u-boot/
git checkout v2018.03 -b tmp
```

Patches:

~/u-boot

```
wget -c
https://rcn-ee.com/repos/git/u-boot-patches/v2018.03/0001-am335x_evm-uEnv.
txt-bootz-n-fixes.patch
wget -c
https://rcn-ee.com/repos/git/u-boot-patches/v2018.03/0002-U-Boot-BeagleBon
e-Cape-Manager.patch

patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
patch -p1 < 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

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-v4.4 (Longterm 4.4.x):

~/bb-kernel/

```
git checkout origin/am33x-v4.4 -b tmp
```

For am33x-rt-v4.4 (Longterm 4.4.x + Real-Time Linux):

~/bb-kernel/

```
git checkout origin/am33x-rt-v4.4 -b tmp
```

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-v4.15 (Stable):

~/bb-kernel/

```
git checkout origin/am33x-v4.15 -b tmp
```

For am33x-v4.16 (Prepatch):

~/bb-kernel/

```
git checkout origin/am33x-v4.16 -b tmp
```

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 v4.4.x:

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-4.4.y -b tmp
```

For TI v4.4.x: Real-Time

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-rt-4.4.y -b tmp
```

For TI v4.9.x:

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-4.9.y -b tmp
```

For TI v4.9.x: Real-Time

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-rt-4.9.y -b tmp
```

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
```

Root File System

Debian 9

User	Password
debian	temppwd
root	root

Download:

~/

```
wget -c
https://rcn-ee.com/rootfs/eewiki/minfs/debian-9.3-minimal-armhf-2017-12-09
.tar.xz
```

Verify:

~/

```
sha256sum debian-9.3-minimal-armhf-2017-12-09.tar.xz
5120fcfb8ff8af013737fae52dc0a7ecc2f52563a9aa8f5aa288aff0f3943d61
debian-9.3-minimal-armhf-2017-12-09.tar.xz
```

Extract:

~/

```
tar xf debian-9.3-minimal-armhf-2017-12-09.tar.xz
```

Ubuntu 16.04 LTS

User	Password
ubuntu	temppwd

Download:

~/

```
wget -c
https://rcn-ee.com/rootfs/eewiki/minfs/ubuntu-16.04.3-minimal-armhf-2017-1
2-09.tar.xz
```

Verify:

~/

```
sha256sum ubuntu-16.04.3-minimal-armhf-2017-12-09.tar.xz
cacc1a8c56649808e3bc27ca58f94dbe817b9e86e660780009a3535709823cc1
ubuntu-16.04.3-minimal-armhf-2017-12-09.tar.xz
```

Extract:

~/

```
tar xf ubuntu-16.04.3-minimal-armhf-2017-12-09.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/
```

Dealing with old Bootloader in eMMC

If you don't want to clear out the old Bootloader in eMMC add this uEnv.txt to /media/rootfs/

~/uEnv.txt

```
##This will work with: Angstrom's 2013.06.20 u-boot.

loadaddr=0x82000000
fdtaddr=0x88000000
rdaddr=0x88080000

initrd_high=0xffffffff
fdt_high=0xffffffff

#for single partitions:
mmcroot=/dev/mmcblk0p1

loadximage=load mmc 0:1 ${loadaddr} /boot/vmlinuz-${uname_r}
loadxfdt=load mmc 0:1 ${fdtaddr} /boot/dtbs/${uname_r}/${fdtfile}
loadxrd=load mmc 0:1 ${rdaddr} /boot/initrd.img-${uname_r}; setenv rdsize
${filesize}
loaduEnvtxt=load mmc 0:1 ${loadaddr} /boot/uEnv.txt ; env import -t
${loadaddr} ${filesize};
loadall=run loaduEnvtxt; run loadximage; run loadxfdt;

mmcargs=setenv bootargs console=tty0 console=${console} ${optargs}
${cape_disable} ${cape_enable} root=${mmcroot} rootfstype=${mmcrootfstype}
${cmdline}

uenvcmd=run loadall; run mmcargs; bootz ${loadaddr} - ${fdtaddr};
```

~/

```
sudo cp -v ./uEnv.txt /media/rootfs/
```

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/${kernel_version}.zImage
/media/rootfs/boot/vmlinuz-${kernel_version}
```

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/boot/dtbs/${kernel_version}/
```

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/rootfs/etc/fstab"
```

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}=="1", KERNEL=="eth*", NAME="eth0"
```

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-get update
sudo apt-get 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/
eMMC/bbb-eMMC-flasher-eewiki-ext4.sh
chmod +x bbb-eMMC-flasher-eewiki-ext4.sh
sudo /bin/bash ./bbb-eMMC-flasher-eewiki-ext4.sh
```

SGX

Build SGX modules/userspace (must be done on an x86, due to the TI 5.01.01.02 blob extractor)

~/bb-kernel/

```
./sgx_build_modules.sh
```

Copy ./deploy/GFX_5.01.01.02.tar.gz to BeagleBone/BeagleBone Black and install

```
sudo tar xfv GFX_5.01.01.02.tar.gz -C /
cd /opt/gfxinstall/
sudo ./sgx-install.sh
sudo reboot
```

Verify omaplfb & pvrsrvkm loaded

```
debian@arm:~$ lsmod | grep omaplfb
omaplfb                 12065  0
pvrsrvkm               178782  1 omaplfb
```

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/<file0>.dtbo
uboot_overlay_addr1=/lib/firmware/<file1>.dtbo
uboot_overlay_addr2=/lib/firmware/<file2>.dtbo
uboot_overlay_addr3=/lib/firmware/<file3>.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/<file4>.dtbo
uboot_overlay_addr5=/lib/firmware/<file5>.dtbo
uboot_overlay_addr6=/lib/firmware/<file6>.dtbo
uboot_overlay_addr7=/lib/firmware/<file7>.dtbo
```

U-Boot: PRU Options (v4.4.x-ti)

/boot/uEnv.txt

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

U-Boot: PRU Options (v4.4.x-ti & v4.x.x-bone)

/boot/uEnv.txt

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

/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