BeagleBone

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

Availability

Boards:
BeagleBone at Digi-Key

Basic Requirements

- Running a recent supported release of Debian, Fedora or Ubuntu on a x86 64bit based PC; without OS Virtualization Software.
- Many of the listed commands assume /bin/bash as the default shell.
- ARM Cross Compiler – Linaro: https://www.linaro.org
  - Linaro Toolchain Binaries: https://www.linaro.org/downloads/
- Bootloader
  - Das U-Boot – the Universal Boot Loader: http://www.denx.de/wiki/U-Boot
  - Source: https://github.com/u-boot/u-boot/
- Linux Kernel
  - Linus's Mainline tree: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
- ARM based rootfs
  - Debian: https://www.debian.org
  - Ubuntu: https://www.ubuntu.com

ARM Cross Compiler: GCC

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

user@localhost:~$

```
wget -c https://releases.linaro.org/components/toolchain/binaries/6.5-2018.12/arm-linux-gnueabihf/gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabihf.tar.xz
tar xf gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabihf.tar.xz
export CC=`pwd`/gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

Test Cross Compiler:

user@localhost:~$

```
${CC}gcc --version
```

Test Output:

```
arm-linux-gnueabihf-gcc (Linaro GCC 6.5-2018.12) 6.5.0
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:

user@localhost:~$

```
git clone -b v2019.04 https://github.com/u-boot/u-boot --depth=1
cd u-boot/
```

Patches:

user@localhost:~/u-boot$

```
wget -c https://github.com/eewiki/u-boot-patches/raw/master/v2019.04/0001-am335x_evm-uEnv.txt-bootz-n-fixes.
patch
wget -c https://github.com/eewiki/u-boot-patches/raw/master/v2019.04/0002-U-Boot-BeagleBone-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:

user@localhost:~/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-v4.14 (Longterm 4.14.x):

user@localhost:~/bb-kernel$

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

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

user@localhost:~/bb-kernel$

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

For am33x-v4.19 (Longterm 4.19.x):

user@localhost:~/bb-kernel$

```
git checkout origin/am33x-v4.19 -b tmp
```

For am33x-rt-v4.19 (Longterm 4.19.x + Real-Time Linux):

user@localhost:~/bb-kernel$

```
git checkout origin/am33x-rt-v4.19 -b tmp
```

For am33x-v5.4 (Longterm 5.4.x):

user@localhost:~/bb-kernel$

```
git checkout origin/am33x-v5.4 -b tmp
```

For am33x-rt-v5.4 (Longterm 5.4.x + Real-Time Linux):

user@localhost:~/bb-kernel$

```
git checkout origin/am33x-rt-v5.4 -b tmp
```

Build:

user@localhost:~/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
```

For TI v4.19.x:

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-4.19.y -b tmp
```

For TI v4.19.x: Real-Time

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-rt-4.19.y -b tmp
```

For TI v5.4.x:

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-5.4.y -b tmp
```

For TI v5.4.x: Real-Time

~/ti-linux-kernel-dev/

```
git checkout origin/ti-linux-rt-5.4.y -b tmp
```

Build:

user@localhost:~/ti-linux-kernel-dev$

```
./build_kernel.sh
```

Root File System

Debian 10

User	Password
debian	temppwd
root	root

Download:

user@localhost:~$

```
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/debian-10.4-minimal-armhf-2020-05-10.tar.xz
```

Verify:

user@localhost:~$

```
sha256sum debian-10.4-minimal-armhf-2020-05-10.tar.xz
```

sha256sum output:

```
cd598e42850cbef87602bf15ee343abfbf0d8c6ba81028c741672b5f24263534  debian-10.4-minimal-armhf-2020-05-10.tar.xz
```

Extract:

user@localhost:~$

```
tar xf debian-10.4-minimal-armhf-2020-05-10.tar.xz
```

Ubuntu 20.04 LTS

User	Password
ubuntu	temppwd

Download:

user@localhost:~$

```
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/ubuntu-20.04-minimal-armhf-2020-05-10.tar.xz
```

Verify:

user@localhost:~$

```
sha256sum ubuntu-20.04-minimal-armhf-2020-05-10.tar.xz
```

sha256sum output:

```
de0177ac9259fdbcc626ee239f4258b64070c0921dbc38c45fab6925a5becaa1  ubuntu-20.04-minimal-armhf-2020-05-10.tar.xz
```

Extract:

user@localhost:~$

```
tar xf ubuntu-20.04-minimal-armhf-2020-05-10.tar.xz
```

Setup microSD card

We need to access the External Drive to be utilized by the target device. Run lsblk to help figure out what linux device has been reserved for your External Drive.

Example: for DISK=/dev/sdX

```
lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 465.8G  0 disk
sda1   8:1    0   512M  0 part /boot/efi
sda2   8:2    0 465.3G  0 part /               <- Development Machine Root Partition
sdb      8:16   1   962M  0 disk                 <- microSD/USB Storage Device
sdb1   8:17   1   961M  0 part                 <- microSD/USB Storage Partition
```

Thus you would use:

```
export DISK=/dev/sdb
```

Example: for DISK=/dev/mmcblkX

```
lsblk
NAME      MAJ:MIN   RM   SIZE RO TYPE MOUNTPOINT
sda         8:0      0 465.8G  0 disk
sda1      8:1      0   512M  0 part /boot/efi
sda2      8:2      0 465.3G  0 part /               <- Development Machine Root Partition
mmcblk0     179:0    0   962M  0 disk                 <- microSD/USB Storage Device
mmcblk0p1 179:1    0   961M  0 part                 <- microSD/USB Storage Partition
```

Thus you would use:

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

Erase partition table/labels on microSD card:

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

Install Bootloader:

user@localhost:~$

```
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.

Check the version of sfdisk installed on your pc

```
sudo sfdisk --version
```

Example Output

```
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:

```
for: DISK=/dev/mmcblkX
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 be auto-mounted...

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

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

for: DISK=/dev/sdX
sudo mount ${DISK}1 /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=5.X.Y-Z]
----------------------------
```

Copy and paste that "export kernel_version=5.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=5.X.Y-Z
```

Copy Root File System

user@localhost:~$

```
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

user@localhost:~$

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

Copy Kernel Image

Kernel Image:

user@localhost:~$

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

Copy Kernel Device Tree Binaries

user@localhost:~$

```
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

user@localhost:~$

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

File Systems Table (/etc/fstab)

user@localhost:~/$

```
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
```

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: 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 (v4.14.x-ti)

/boot/uEnv.txt

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

U-Boot: PRU Options (v4.19.x-ti)

/boot/uEnv.txt

```
uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-19-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

Any questions or comments please go to our TechForum: TechForum