

# OSD3358-SM-RED

This is a page about Octavo System's TI Cortex-A8 based; OSD3358-SM-RED.

- Availability
- Basic Requirements
- ARM Cross Compiler: GCC
- Bootloader: U-Boot
- Linux Kernel
  - Mainline
  - TI BSP
- Root File System
  - Debian 9
  - Ubuntu 18.04 LTS
- Setup microSD card
  - Backup Bootloader
- 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
  - U-Boot Overlays
- Comments

## Availability

Boards:

OSD3358-SM-RED 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: <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/cgit/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 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/6.5-2018.12/arm-linux-gnueabi/f/gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabi/f.tar.xz
tar xf gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabi/f.tar.xz
export CC=`pwd`/gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabi/f/bin/arm-linux-gnueabi/f-
```

Test Cross Compiler:

```
~/
```

```
`${CC}gcc --version
arm-linux-gnueabi/f-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:

```
~/
```

```
git clone https://github.com/u-boot/u-boot
cd u-boot/
git checkout v2019.04 -b tmp
```

Patches:

```
~/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:

```
~/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):

```
~/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-v4.19 (Longterm 4.19.x):

```
~/bb-kernel/
```

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

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

```
~/bb-kernel/
```

```
git checkout origin/am33x-rt-v4.19 -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 v4.14.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
```

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/minifs/debian-9.8-minimal-armhf-2019-02-16.tar.xz
```

Verify:

```
~/
```

```
sha256sum debian-9.8-minimal-armhf-2019-02-16.tar.xz  
40643313dbfc4bc9487455cb6e839cc110e226ac2e9046a2f59f05e563802943  debian-  
9.8-minimal-armhf-2019-02-16.tar.xz
```

Extract:

```
~/
```

```
tar xf debian-9.8-minimal-armhf-2019-02-16.tar.xz
```

### Ubuntu 18.04 LTS

User	Password
ubunt u	temppwd

Download:

```
~/  
wget -c https://rcn-ee.com/rootfs/eewiki/minifs/ubuntu-18.04.2-minimal-  
armhf-2019-02-16.tar.xz
```

Verify:

```
~/  
sha256sum ubuntu-18.04.2-minimal-armhf-2019-02-16.tar.xz  
299f278ff446f0e427cb1a6650d1fa5d42d6744fff627c804f0afc918129b6b6  ubuntu-  
18.04.2-minimal-armhf-2019-02-16.tar.xz
```

Extract:

```
~/  
tar xf ubuntu-18.04.2-minimal-armhf-2019-02-16.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:

~/

```
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:

```
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 will 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/
```

## 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/u-boot/
sudo cp -v ./u-boot/MLO /media/rootfs/opt/backup/u-boot/
sudo cp -v ./u-boot/u-boot.img /media/rootfs/opt/backup/u-boot/
```



## 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: ctrl-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/eMMC/bbb-eMMC-flasher-eewiki-ext4.sh
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/<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](#)