

Zen Cape Audio Guide – Kernel 5.10; Image 2022-11-01+

by Brian Fraser
Last update: Feb 27, 2024

Guide has been tested on
BeagleBone (Target): **Debian 11.4**
PC OS (host): **Debian 11.5**

- This document guides the user through**
1. Getting audio output from the Zen cape
 2. Getting audio output from a USB Audio Adapter
 3. Playing PCM (wave) files via a C program through `asound`

Table of Contents

1. Zen Cape Audio: Installing Virtual Audio Cape.....	2
1.1 Accessing the Cape Manager.....	2
2. (Optional) Setting up a USB Audio Adapter.....	6
3. Play Audio.....	7
4. Other Tools Command-line Tools.....	9
4.1 Recording (Untested).....	9
4.2 MP3 Player.....	9
4.3 Text to Speech.....	9
5. Cross-compile C Program to Play PCM Audio.....	10
5.1 Cross-compiling ALSA using CMake.....	11
6. (BROKEN) Recovering from Corrupted uEnv.txt.....	12

Formatting

1. Commands for the host Linux’s console are show as:
`(host)$ echo "Hello PC world!"`
2. Commands for the target (BeagleBone) Linux’s console are shown as:
`(bbg)$ echo "Hello embedded world!"`
3. Almost all commands are case sensitive.

Revision History

- Nov 1, 2022: Updated to Debian Bullseye and use of USB Audio Adapter
- Feb 9, 2023: Installing the audio cape no longer disables the universal cape.
- Feb 27, 2024: Install `armhf` library on the host using `apt`.

1. Zen Cape Audio: Installing Virtual Audio Cape

For Linux to use the audio hardware on the Zen cape, we must install a device tree file to tell the kernel how to access the hardware.

If a guide you find uses the `$/SLOTS` file or the “cape manager” then that guide is out of date and does not apply to kernel 4.9+.

1.1 Accessing the Cape Manager

Linux must be told what hardware is connected to the CPU. It learns this at boot up using a Device Tree (file is a .DTB for the device tree binary). The boot loader (UBoot) detects what capes are installed (or configured) and sets up the device tree for the kernel to use.

Do the following just once (per board).

1. Backup `uEnv.txt`

The `uEnv.txt` file is critical to controlling how UBoot starts the system. We will change it to load the audio setup; however, we must first take a backup copy to recover from some errors.

```
(bbg) $ cd /boot
(bbg) $ sudo cp uEnv.txt uEnv-BeforeAudio.txt
```

- **WARNING:** Corrupting the `uEnv.txt` file may cause the BeagleBone to be unbootable, and hence must be reflashed. Be careful editing this file!

2. Edit `uEnv.txt` to load the Audio overlays

- Edit `uEnv.txt`:

```
(bbg) $ sudo nano /boot/uEnv.txt
```

- Find the section titled:

```
###Additional custom capes
```

- Change it to be:

```
###Additional custom capes
uboot_overlay_addr4=/lib/firmware/BB-BONE-AUDI-02-00A0.dtbo
#uboot_overlay_addr5=<file5>.dtbo
#uboot_overlay_addr6=<file6>.dtbo
#uboot_overlay_addr7=<file7>.dtbo
```

- The first line (`BB-BONE-AUDI`) loads the audio device tree overlay.
- Ensure you **removed the #** on the line to uncomment it.

3. Reboot the target.

4. Verify it worked:

- Ensure the audio has loaded (first give it access to the internet):

```
(bbg) $ sudo apt install alsa-utils
```

```
...
```

```
(bbg) $ aplay -l # lower-case L
```

```
**** List of PLAYBACK Hardware Devices ****
```

```
card 0: B [AudioCape Rev B], device 0: davinci-mcasp.0-tlv320aic3x-hifi
```

```
tlv320aic3x-hifi-0 [davinci-mcasp.0-tlv320aic3x-hifi tlv320aic3x-hifi-0]
```

```
Subdevices: 1/1
```

```
Subdevice #0: subdevice #0
```

- Ensure the I²C for bus 1 has still loaded:

```
(bbg)$ config-pin p9_18 i2c
(bbg)$ config-pin p9_17 i2c
(bbg)$ i2cdetect -l
i2c-1 i2c          OMAP I2C adapter          I2C adapter
i2c-2 i2c          OMAP I2C adapter          I2C adapter
i2c-0 i2c          OMAP I2C adapter          I2C adapter
```

```
(bbg)$ i2cdetect -y -r 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  1c  --  --  --
20: 20  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

If this `i2cdetect` command runs very slowly (> 1s) then you likely have not run `config-pin` on the two I²C pins.

If you are using a Zen Cape red, then you may not see “1c” but rather “18” in the figure; that is fine.

5. Troubleshooting

- If your board does not boot after editing the `uEnv.txt` file then you likely need to reflash your board. There used to be a process to restore `uEnv.txt` from your backup; however that no-longer works.
- If your board boots, but the audio adapter does not load, double check your modifications to the `/boot/uEnv.txt` file.
- If Linux pauses a long time during boot and then prints the following error on the serial port, then the device overlay is likely incompatible with our version of the BeagleBone image. Reflash your board to a newer version of the image.

```
...
Gave up waiting for root file system device. Common problems:
- Boot args (cat /proc/cmdline)
  - Check rootdelay= (did the system wait long enough?)
  - Missing modules (cat /proc/modules; ls /dev)
ALERT! /dev/mmcblk1p1 does not exist. Dropping to a shell!
```

```
BusyBox v1.22.1 (Debian 1:1.22.0-19+b3) built-in shell (ash)
Enter 'help' for a list of built-in commands.
```

```
(initramfs)
```

- When a custom cape is loaded (such as the audio one), the universal cape can sometimes be disabled. This means that fewer GPIO pins are automatically exported, and the `config-pin` utility will not work on them. See GPIO guide for exporting pins.
- General debugging of device tree issues:

- Use the serial port (via the Zen cape) to view the messages that UBoot prints out during startup. If your kernel is booting, then also have a good look for messages in `dmesg` as it may tell you what goes wrong after UBoot configures the system.

Using `screen`, capture the output to the file `~/screenlog.0` via: CTRL a, H
(Press control, press ‘a’; release control, press SHIFT, press ‘h’)

- When you don’t specify any specific capes to load, the BeagleBone automatically loads a number of virtual capes:
 - a. `am335x-boneblack-uboot.dtb`: The full base device tree binary file. This is not an “overlay”: it is a full configuration file which other overlays can be loaded on-top of. It is found in `/boot/dtbs/<kernel version>/`
 - b. `BB-BONE-eMMC1-01-00A0.dtbo`: overlay which loads support for the build-in eMMC. If this is not correctly loaded then your board will not boot from the onboard eMMC!
 - c. `BB-ADC-00A0.dtbo`: overlay which loads support for the analog to digital converter.
 - d. `univ-bbb-Exx-00A0.dtbo`: “universal cape” overlay which maps many otherwise unused pins to be GPIO and available for a “maker” to use in projects. Pins mapped by the universal cape can be configured using the `configure-pin` utility. Note that the `Exx` means that it’s loading support for eMMC, but neither HDMI video nor HDMI audio. If overlays are explicitly loaded, the universal cape is not loaded (otherwise it is likely to interfere with the pin configuration for the cape).
 - e. Depending on the configuration of your ZEN cape, UBoot may also detect the `BB-BONE-ZEN-01` cape. However, there is no overlay file for this, so it is ignored by UBoot. You can ignore any error related to not finding this overlay.
- You can check if the universal cape is loaded by seeing what pins are mapped to GPIO by default.

```
(bbg) $ ls /sys/class/gpio
```

If it shows only a few (my board has just two `gpio##/` folders), then the universal cape is not loaded.

Also try (for `P9_28`, as an example):

```
(bbg) $ ls /proc/device-tree/ocp/P9_28_pinmux
```

```
(bbg) $ ls /proc/device-tree/ocp/cape-universal/P9_28
```

If neither of these commands find any entries, then the universal cape is likely not loaded.

- Device tree overlays are stored in `/lib/firmware`. You can recover the source file from a binary file for a device tree (`.dtb` or `.dtbo`):

```
(bbg) $ dtc -I dtb -O dts /lib/firmware/BB-BONE-AUDI-02-00A0.dtbo
```

- Useful Links:

- a. When loading, if you get an error in UBoot, the following may shed light on what the error codes mean:

<https://elixir.bootlin.com/linux/latest/source/scripts/dtc/libfdt/libfdt.h#L16>

- b. Latest source on BeagleBone device tree overlays:

<https://github.com/beagleboard/bb.org-overlays>

- c. Information on the `uEnv.txt` file:
https://elinux.org/Beagleboard:BeagleBoneBlack_Debian#U-Boot_Overlays
- d. Device Trees and the Linux kernel:
<https://www.kernel.org/doc/Documentation/devicetree/usage-model.txt>
https://elinux.org/Device_Tree_Reference
- e. If RobertCNelson wrote it, it's probably correct.

2. (Optional) Setting up a USB Audio Adapter

If you are using a USB audio adapter (such as an [ENVEL USB to 3.5mm Audio Adapter](#)), the following steps will configure the board to be able to play audio. Note that not all USB audio adapters will work: I purchased four from Amazon and only 2 functioned with the others preventing the board from turning on, booting correctly, or loading a driver.

1. On the target, install the sound tools and libraries (aplay, alsamixer, lsusb)

```
(bbg)$ sudo apt-get install usbutils alsa-utils i2c-tools
```

2. Plug in your USB Audio Adapter to your target. Ensure it is detected:

```
(bbg)$ lsusb
```

Possible output:

```
Bus 001 Device 003: ID 1b3f:2008 Generalplus Technology Inc. USB Audio Device
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

- If needed, run `dmesg` to find out information about how the Linux kernel tried to load drivers, etc.
- This guide tested with:
 - “Generalplus Technology Inc. USB Audio Device” from Envel
 - “C-Media Electronics, Inc. Audio Adapter (Unitek Y-247A)” from Cable Creation

3. Set the default sound card for ALSA on the target:

- View sound cards available:

```
(bbg)$ cat /proc/asound/cards
```

Output:

```
1 [Device          ]: USB-Audio - USB Audio Device
    GeneralPlus USB Audio Device at usb-musb-hdrc.1-1, full speed
```

- Set the default, changing the “1” below to the desired device number listed above:

```
(bbg)$ sudo nano /etc/asound.conf
```

Set contents to:

```
defaults.pcm.card 1
defaults.ctl.card 1
```

3. Play Audio

1. On the target, install the sound tools and libraries (aplay, alsamixer, lsusb)
(bbg)\$ sudo apt install usbutils alsa-utils i2c-tools
2. Plug in speakers or headphones into the audio output on the Zen cape (green 3.5mm socket).
 - **WARNING:** When testing your audio, do not put headphones in your ear. A very loud sound is possible if there are problems, and this could cause an injury to your ear. Just drape the head-phones beside your ears so you can hear it, but not be injured if it goes wrong. Once you know the audio levels are fine, then using head-phones normally is fine.
3. Save a WAVE file to your NFS folder on your host (say, sample.wav).
4. On the target, play the file with:
(bbg)\$ **aplay sample.wav**
 - My board sometimes has issues playing files directly from the NFS folder; try copying the wave file to your home folder on the target first. This sometimes requires a reboot for me.
5. Change the volume:
(bbg)\$ **alsamixer**
 - Use the left/right arrows to select different channels to adjust.
 - Use the up/down arrows to change the volume.
 - Press 'M' to mute or unmute channels.
 - Press ESC to exit (and save changes).
 - Change the volume of wave data playback by changing the PCM channel.
6. Troubleshooting:
 - You can list available playback devices to ensure the configuration succeeded:
(bbg)\$ **aplay -l**
**** List of PLAYBACK Hardware Devices ****
card 0: B [AudioCape Rev B], device 0: davinci-mcasp.0-tlv320aic3x-hifi
tlv320aic3x-hifi-0 []
Subdevices: 1/1
Subdevice #0: subdevice #0
 - You can also list PCM playback devices:
(bbg)\$ **aplay -L**
 - You can view some info:
(bbg)\$ **cat /proc/asound/cards**
 - You can play back white noise to test the hardware:
(bbg)\$ **speaker-test**
 - You can playback a sound on a specific card using a command such as:
(bbg)\$ **aplay -D default:CARD=Device ./my-file.wav**
Where the `CARD` parameter can be found by running ``aplay -L``, look for “`CARD=`”
 - If the output from the software looks like it should be playing audio but no sound is generated, ensure you have speakers or headphones plugged into the green audio-out jack, and that you have fully pushed in the connector (may hear a small click as it goes in all the way).
 - If you get the following error with `alsamixer`, it likely means the audio cape is not loaded:
cannot open mixer: No such file or directory
 - If you get the error:
Unable to set hw params for playback: Invalid argument

or

```
ALSA lib confmisc.c:768:(parse_card) cannot find card '0'
```

It likely means your device tree is incorrect; see section 1: Zen Cape Audio: Installing Virtual Audio Cape

- If you get the following error with `aplay` while using a USB audio device, it likely means you have not yet set your default audio device, or your USB Audio Adapter is not connected:

```
ALSA lib confmisc.c:767:(parse_card) cannot find card '0'
```

```
ALSA lib conf.c:4745:(_snd_config_evaluate) function snd_func_card_driver  
returned error: No such file or directory
```

```
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
```

```
ALSA lib conf.c:4745:(_snd_config_evaluate) function snd_func_concat  
returned error: No such file or directory
```

```
ALSA lib confmisc.c:1246:(snd_func_refer) error evaluating name
```

```
ALSA lib conf.c:4745:(_snd_config_evaluate) function snd_func_refer  
returned error: No such file or directory
```

```
ALSA lib conf.c:5233:(snd_config_expand) Evaluate error: No such file or  
directory
```

```
ALSA lib pcm.c:2660:(snd_pcm_open_noupdate) Unknown PCM default
```

```
aplay: main:830: audio open error: No such file or directory
```

- If using a USB audio device, if you see an error like:
"simple.c:283:snd_mixer_selem_get_playback_volume_range: Assertion 'elem' failed."
Check to ensure you have setup `/etc/asound.conf` correctly with no typos.

4. Other Tools Command-line Tools

4.1 Recording (Untested)

1. Record with:

```
(bbg) $ arecord -r 44100 -f S16_LE -c 2 testRecording.wav
```

2. Things to do to prove out the recording capabilities more:

- Types of microphones, and mic settings need to be investigated.
- Volume controls for recording need to be investigated.
- Tested using an audio cable from Zen headphone jack back into Zen mic port. Able to record poor quality quiet audio using the above command. Recommend using an audio file processing tool (such as GoldWave for Windows) to view recordings while debugging.

4.2 MP3 Player

1. Install the `mpg123` package:

```
(bbg) $ sudo apt-get update
```

```
(bbg) $ sudo apt-get install mpg123
```

- For this to work, you must have internet access. Test by pinging Google.
- Note that installing this pulls in a number of other packages at the same time.

2. Copy an MP3 file to your NFS share.

3. Play the MP3:

```
(bbg) $ mpg123 sample.mp3
```

- You can view the amount of CPU consumed during playback by loading a new terminal to the target:

```
(bbg) $ top
```

This will show you the CPU usage of `mplayer` (~5% on my test).

- You can also use other command-line MP3 players such as `mplayer` (`apt-get install it`). Takes up ~250 MB to install its libraries.

4.3 Text to Speech

1. Install the text-to-speech engine:

```
(bbg) $ sudo apt-get update
```

```
(bbg) $ sudo apt install espeak
```

2. Generate a wave file:

```
(bbg) $ espeak 'All your bits are belong to us.' -w test.wav
```

3. Playback the audio:

```
(bbg) $ aplay test.wav
```

5. Cross-compile C Program to Play PCM Audio

1. On the *BeagleBone*, install `asound`

```
(bbg) $ sudo apt update
(bbg) $ sudo apt install libasound2
```

- You can check for the necessary files using:
(bbg) \$ `ls /usr/lib/arm-linux-gnueabi/hf/libasound*`
`libasound.so.2 libasound.so.2.0.0`

2. On the *host*, install the `asound` development library (for the header files)

```
(host) $ sudo apt update
(host) $ sudo apt install libasound2-dev
```

3. The *host* will need the `armhf` version of the `asound` library files in order to cross-compile the application. To do this we will install the `armhf` library on the host. (Very useful for cross-compiling libraries!)

- Setup the host to be able to install `armhf` files and install ALSA:

```
(host) $ sudo dpkg --add-architecture armhf
(host) $ sudo apt update
(host) $ sudo apt install libasound2-dev:armhf
```

- Verify the library installed:

```
(host) $ ls /usr/lib/arm-linux-gnueabi/hf/libasound.so
/usr/lib/arm-linux-gnueabi/hf/libasound.so
```

4. Download the `wave_player.c` example from the course website.

- In a directory on your host, such as in `~/cmpt433/work/pcmExample/`, copy the `wave_player.c` and `Makefile`.
- Create a `wave-files/` sub-directory of this folder, and copy a wave file into it. For example, copy in the provided drum sounds wave files.
- Note that the program assumes the files are 16-bit, signed little endian, 44.1kHz, mono files (which is true of the drum sounds). If your sounds are different, you'll need to change the settings in `wave_player.c`

5. Cross-compile the example code by running `make`:

```
(host) $ cd ~/cmpt433/work/pcmExample/
(host) $ make
```

- `Makefile` will build the `wave_player.c` code into `wave_player` and place it in `~/cmpt433/public/myApps/`.
- It links against the locally installed (on the host) `armhf` library which it will find automatically in `/usr/lib/arm-linux-gnueabi/hf/`
- The `Makefile` includes the `-lasound` flag, which is needed for the compiler to link against the `libasound.so` library.
- The `wav` target in the `Makefile` copies the `wave-files/` folder into the `myApps/` folder.
- See comments in `wave_player.c` for details on how application works.

6. Run the application:

```
(bbg) $ cd /mnt/remote/myApps/
(bbg) $ wave_player
```

- You should hear the drum sound selected in the `.c` file. The drum sounds are quite short.
- For reference, the part of the application which actually sends data to be played is the call to `snd_pcm_writei()` in the `Audio_playFile()` function.

This call is blocking: it waits until the data has been transmitted to the ALSA sub-system for playback. However, there is some hardware buffering, so the sound may not have actually stopped when `snd_pcm_writei()` returns.

You can use this delay to send more data, hopefully fast enough so that the sound has no jitter. Or, if you want to exit, you may want to call `snd_pcm_drain()` first so that all buffers play out without clipping the end of your wave file.

7. Troubleshooting:

- When trying to compile, if you get the following error:
`fatal error: alsa/asoundlib.h: No such file or directory`
Ensure you have `libasound2-dev` installed on your host PC.
- When running `wave_player`, if you see:
`error while loading shared libraries: libasound.so.2: cannot open shared object file: No such file or directory`
Then you likely need to install `libasound2` on the target.
- If you don't hear any sound when running `wave_player`, use `aplay` to ensure your hardware is configured correctly and your mixer level is set correctly.
- If you see an error:
`ALSA lib confmisc.c:768:(parse_card) cannot find card '0'`
it likely means that you don't have the `BB-BONE-AUDI-02` cape loaded; see previous section for loading the audio cape.

5.1 Cross-compiling ALSA using CMake

The website includes a sample project to cross-compile the above application using CMake. It includes the following changes to `CMakeList.txt` for the component that using the ALSA code (such as your main application):

1. Declare that it uses the ALSA library (will automatically find the `armhf` library installed in the above steps):

```
# ALSA support
find_package(ALSA REQUIRED)
target_link_libraries(wave_player_cmake LINK_PRIVATE asound)
```
2. Copy the wave files folder to your NFS folder:

```
# Copy the folder of WAVE files to NFS
add_custom_command(TARGET wave_player_cmake POST_BUILD
  COMMAND "${CMAKE_COMMAND}" -E copy_directory
    "${CMAKE_SOURCE_DIR}/wave-files"
    "~/cmpt433/public/myApps/wave-files"
  COMMENT "Copying WAVE files to public NFS directory")
```

Change “`wave_player_cmake`” to be the name of your executable.

6. (BROKEN) Recovering from Corrupted uEnv.txt

The latest version of the BBG firmware (for example, Bullseye 2022-11-01) change how the eMMC partitions are stored. This makes our version of UBoot unable to write to drive. This prevents any recovery processes described below. The best option is often to reflash the board.

If you edit `/boot/uEnv.txt` and it becomes corrupted, or you load a device tree which does not support the onboard eMMC then your board may fail to boot. These steps should help you recover.

1. View your board's boot process using the serial port on the board (via the `screen` program).
2. Reboot your board. You should see:

```
U-Boot SPL 2018.01-00002-g9aa111a004 (Jan 20 2018 - 12:45:29)
Trying to boot from MMC2

U-Boot 2018.01-00002-g9aa111a004 (Jan 20 2018 - 12:45:29 -0600), Build:
jenkins-github_Bootloader-Builder-32

CPU   : AM335X-GP rev 2.1
I2C:   ready
DRAM:  512 MiB
No match for driver 'omap_hsmmc'
No match for driver 'omap_hsmmc'
Some drivers were not found
Reset Source: Global external warm reset has occurred.
Reset Source: Power-on reset has occurred.
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment

Board: BeagleBone Black
<ethaddr> not set. Validating first E-fuse MAC
BeagleBone Black:
Model: SeeedStudio BeagleBone Green:
debug: process_cape_part_number:[BB-BONE-ZEN-01]
debug: process_cape_part_number:[42422D424F4E452D5A454E2D3031]
BeagleBone: cape eeprom: i2c_probe: 0x54: /lib/firmware/BB-BONE-ZEN-01-
00A0.dtbo [0xeb9aeff]
BeagleBone: cape eeprom: i2c_probe: 0x55:
BeagleBone: cape eeprom: i2c_probe: 0x56:
BeagleBone: cape eeprom: i2c_probe: 0x57:
Net:   eth0: MII MODE
cpsw, usb_ether
Press SPACE to abort autoboot in 2 seconds
=>
```

3. Press SPACE as soon as it begins booting to enter the UBoot prompt.

4. Copy a file, changing SOURCE and TARGET as needed:

```
=> ext4load mmc 1:1 0x82000000 SOURCE
=> ext4write mmc 1:1 0x82000000 TARGET ${filesize}
```

- For example, to make a backup copy of your current `uEnv.txt` use:

```
=> ext4load mmc 1:1 0x82000000 /boot/uEnv.txt
=> ext4write mmc 1:1 0x82000000 /boot/uEnv.bak.uboot ${filesize}
```
- For example, to restore `/boot/uEnv-BeforeAudio.txt` use:

```
=> ext4load mmc 1:1 0x82000000 /boot/uEnv-BeforeAudio.txt
=> ext4write mmc 1:1 0x82000000 /boot/uEnv.txt ${filesize}
```

Note the `${filesize}` variable is set when you do an `ext4load` command.

5. Boot the board, which loads `/boot/uEnv.txt`:

```
=> boot
```

6. Troubleshooting:

- When you see the following error message, when trying to write to the onboard eMMC, it likely means that you are running one of the newest versions of the BBG, and it enables some file system checks which prevent UBoot from accessing the drives. At current, there is nothing known to work around this problem other than reflashing the board.

```
=> ext4write mmc 1:1 0x82000000 /boot/uEnv.bak.uboot ${filesize}
Unsupported feature metadata_csum found, not writing.
** Error ext4fs_write() **
** Unable to write file /boot/uEnv.bak.uboot **
=>
```

- You can view the contents of a file using:

```
=> ext4load mmc 1:1 0x82000000 /boot/uEnv.txt
=> md 0x82000000
```

- Listing files in the `/boot/` folder:

```
=> ext4ls mmc 1:1 /boot
```

- When booting, if you see messages from UBoot like:

```
Checking for: /uEnv.txt ...
Checking for: /boot.scr ...
Checking for: /boot/boot.scr ...
Checking for: /boot/uEnv.txt ...
** Invalid partition 2 **
....
```

It likely means you have deleted `/boot/uEnv.txt`

- When booting, if you see messages from UBoot like:

```
mount: can't find /root in /etc/fstab
Target filesystem doesn't have requested /sbin/init.
mount: mounting /dev on /root/dev failed: No such file or directory
No init found. Try passing init= bootarg.
...
BusyBox v1.22.1 (Debian 1:1.22.0-9+deb8u1) built-in shell (ash)
Enter 'help' for a list of built-in commands.
```

```
/bin/sh: can't access tty; job control turned off
(initramfs)
```

It likely means you don't have emmc support correctly loaded in your device tree. Revert to a previous version of `uEnv.txt` (*you do have a backup, right?*) and then change the device tree to be a version which supports the emmc (likely with "emmc" in the file name).

It may also mean that your on-board emmc has been corrupted and needs to be reflashed.