# This document exists in MS Word format because standard ASCII text does not support typographical emphasis (bold, specific fonts, headers, titles, etc.

# Expanding the Beagle Bone Black SD Card Image

## Background

All SD Cards, like hard disk drives have some percentage of "bad sectors"

When BBB SD Images are created, they are purposely created 'very small' so that the image can be placed on any card then exanded as needed.

The steps below, show how to use the BBB to expand the SD Image to fit the size of your SD Card. Alternatively, you could use many of these commands on your Linux machine directly.

Secondly, when the SD Image is expanded we strongly suggest to not fill the entire SD Card to the very last sector.

Reason: It will take a while (several steps) to setup an SD Card. By making the SD Image slightly smaller then your card, when the image is cloned and written to another card it is more likely to fit that other card.

## Overview:

In these steps we will:

* Write the image to an SD Card
* Start with a 4GIG partition.
* Expand this partition to about 7GIG
* Leaving quite a bit of space free.

## Alternate Reading

This document is based on several excellent web resources:

* http://www.kd0cq.com/2014/08/expand-partition-and-create-swap-file-on-beaglebone-black-ubuntu-sdr-img/
* http://elinux.org/Beagleboard:Expanding\_File\_System\_Partition\_On\_A\_microSD
* http://www.berriman.co.uk/bbb-mount-and-shrink-a-disk-image/

## How does this work?

To help understand the process, it is helpful to use a metaphor or an analogy - in this case we'll use a picture frame around your data.

The data on the SD Card is effectively a continuous array of data sectors. The sectors are numbered starting with 0 - ending somewhere around 8Gigabytes (depending on your SD Card).

In sector 0 is what is called an MS-DOS Partition table. This type of partition table can hold between 1 and 4 partitions. You can think of a partition as a "picture frame" around the file system (your data).

Some images have 1 partition; others have 2 or 3 partitions. Find the largest partition because this is where your data is located. Note: If there is a partition after your data, these instructions will not work.

Often data partition does not start at sector 1 - it often starts at some other location. The exact value varies, it is important to know that your card or SD image might use values different then these instructions.

The partition ends "somewhere" at Sector (N).

**Important:** In an earlier step we wrote a new 2G or 4G image onto the SD Card starting at sector 0 (the partition table) - it somewhere near the 2G or 4G location, the remaining ??GIG of space is not allocated.

What we cannot do is "stretch" the partition - that's not possible.

**Instead:** We must delete the existing partition, and then create a new partition that starts at the exact same location as old partition, but ends somewhere near sector X - (depending on your card: 8, 16, 32...GIG)

If there is a partition after your data, your data is boxed or penned in, and cannot be stretched or moved. You’ll need to find another way.

## Step 1 Boot the BBB, and login as ROOT

Boot the BBB from the SD Card

## Step 2 Determine the current data partition, and start location

The goal is to device name that is the root partition, type the command: **df -Th /** The result will look like this:

**$ df -hT /**

**Filesystem Type Size Used Avail Use% Mounted on**

**/dev/mmcblk0p1 ext4 3.3G 2.8G 295M 91% /**

In this example, the image starts a 3.3Gig and is basically full, (only 9%, or 295Mbytes) free – this is why the image must be expanded to the full size of your SD Card. Otherwise the border router cannot build.

**Important Findings:**

* We find the data partition is: **/dev/mmcblk0p1**
* The device as a whole is: **/dev/mmcblk0**
* The **p1** suffix is the naming convention for Partition 1.
* Note: This image has a single partition, other images may additional partitions

## Step 3 Become Root

Become root if not already

**debian@beaglebone:~$ sudo bash**

**Note: default password is: temppwd**

**root@beaglebone:/home/debian#**

## Step 4 Start fdisk

Type the command **fdisk /dev/mmcblk0**

## Step 5 Print the current partition table

The fdisk **p** command prints the current partition table. Note: In this example, there is a single partition. Your image may be configured differently.

**root@beaglebone:/home/debian# fdisk /dev/mmcblk0**

**Welcome to fdisk (util-linux 2.25.2).**

**Changes will remain in memory only, until you decide to write them.**

**Be careful before using the write command.**

**Command (m for help): p**

**Disk /dev/mmcblk0: 7.2 GiB, 7744782336 bytes, 15126528 sectors**

**Units: sectors of 1 \* 512 = 512 bytes**

**Sector size (logical/physical): 512 bytes / 512 bytes**

**I/O size (minimum/optimal): 512 bytes / 512 bytes**

**Disklabel type: dos**

**Disk identifier: 0xca52207f**

**Device Boot Start End Sectors Size Id Type**

**/dev/mmcblk0p1 \* 8192 6963199 6955008 3.3G 83 Linux**

**Important findings:**

* **First:** the Start Sector number is **8192** (we will use this number below).
  + This image, the one and only data partition starts at 8192
  + Your image may have multiple partitions or start at a different location.
  + You are looking for the "Linux" partition.
* **Second:** The End Sector, in this example there is a single partition. Recall the first sector is sector 0, thus we take this value and add **1 (6963199+1) \* 512 = 3565158400**, which is the exact size of the original 'img' file we wrote to the SD CARD.

## Step 6 Delete the partition

**Command (m for help): d**

**Selected partition 1**

**Partition 1 has been deleted.**

**Important:**

* Do not remove the existing ext4 file system signature.
* Newer versions of the FDISK command will ask to remove the existing “ext4” signature
* Do not remove this signature, leave it as is – if this is deleted the image will no longer boot.

## Step 8 Create the new (replacement) partitions

Type the command: **n** to create the replacement partition.

**Important:** When creating the new partition is it critical to use the **SAME FIRST SECTOR** (see above) as the old partition.

This example uses an 8Gig card, thus we use **+7G** for the partition size.

**Command (m for help): n**

**Partition type**

**p primary (0 primary, 0 extended, 4 free)**

**e extended (container for logical partitions)**

**Select (default p): p**

**Partition number (1-4, default 1): 1**

**First sector (2048-15126527, default 2048): 8192**

**Last sector, +sectors or +size{K,M,G,T,P} (8192-15126527, default 15126527): +7G**

**Created a new partition 1 of type 'Linux' and of size 7 GiB.**

## Step 9 To Clone the Card - Determine the last sector

This step is optional and used only if in the future you wish to “clone” a completed (pre-configured) SD Card and distribute it to your co-workers you will need to follow these steps. You do not need to perform this step now (and remember the number below) because you can always run the fdisk command again when needed.

The steps are:

* Run the fdisk command
* Use the p command to print the partition table.

See below:

**Command (m for help): p**

**Disk /dev/mmcblk0: 7.2 GiB, 7744782336 bytes, 15126528 sectors**

**Units: sectors of 1 \* 512 = 512 bytes**

**Sector size (logical/physical): 512 bytes / 512 bytes**

**I/O size (minimum/optimal): 512 bytes / 512 bytes**

**Disklabel type: dos**

**Disk identifier: 0xca52207f**

**Device Boot Start End Sectors Size Id Type**

**/dev/mmcblk0p1 8192 14688255 14680064 7G 83 Linux**

The important number above is **14688255**.

In the future, you can use various tools to read the SD Card and save the image to share others.

**The problem (setup):** These tools read the entire SD card including the free area after the end of the partition up until the last sector of your SD Card.

**The problem (occurs):** When writing to a new SD Card, the last sector of the other card may be different, the full image may, or may not fit on the new card.

**The solution:**  Because (above) we used +7G as the new partition size, the data portion does not run until the very last sector of the SD Card. There exists a few unused sectors (every card will be different). We can thus trim the excess unused sectors such that this image will fit on all SD Cards.

The simplest method is to use the truncate command. Note: The truncate command is a standard Linux tool, and it is also present in the MS-Windows Git Bash distribution of MSYS.

In this example:

* The last partition ends at sector: **14,688,255** (see above)
* Each sector is **512** bytes:
* Remember to add **+1**, to account for sector 0.
* The total size is: **(14,688,255+1) \* 512 = 7,520,387,072**

To truncate this example image use this command:

**truncate -s 7520387072 myimage.img**

## Step 10 Write the partition table to the disk

Use the **w** command to write the partition table to the disk.

**Command (m for help): w**

**The partition table has been altered.**

**Calling ioctl() to re-read partition table.**

**Re-reading the partition table failed.: Device or resource busy**

**The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8).**

## Step 11 Reboot & Error: fsck exited with status code 8.

Some images will have this error, other images will not. To fix this error, do the following:

Reboot the BBB

During the next boot - the BBB might complain as follows:

**Loading, please wait...**

**[ 4.873285] remoteproc1: failed to load am335x-pru0-fw**

**[ 4.918852] remoteproc1: request\_firmware failed: -2**

**[ 4.924046] pru-rproc 4a334000.pru0: rproc\_boot failed**

**[ 5.052414] remoteproc1: failed to load am335x-pru1-fw**

**[ 5.069652] remoteproc1: request\_firmware failed: -2**

**[ 5.074889] pru-rproc 4a338000.pru1: rproc\_boot failed**

**fsck: error 2 (No such file or directory) while executing fsck.ext4 for /dev/mmcblk0p1**

**fsck exited with status code 8**

After a short 20 second pause, the login prompt appears.

For now - Ignore this error, the error will be fixed in a few steps.

## Step 12 - Grow (resize) the actual file system.

In the previous step, and using the "picture frame" analogy - we have removed and replaced the smaller picture frame with a much larger picture frame. However we have not yet made the additional space within the frame available to the file system.

Once the BBB has rebooted, login and become root.

In this example the data partition is: **/dev/mmcblk0p1**, if it where partition 2, the parameter would be **/dev/mmcblk0p2**.

The resize command resizes the filesystem.

**(if needed, "sudo bash" to get the root prompt)**

**root@beaglebone:/home/debian# resize2fs /dev/mmcblk0p1**

**resize2fs 1.43 (17-May-2016)**

**Filesystem at /dev/mmcblk0p1 is mounted on /; on-line resizing required**

**old\_desc\_blocks = 1, new\_desc\_blocks = 1**

**The filesystem on /dev/mmcblk0p1 is now 1835008 (4k) blocks long.**

Note: The resize2fs command can also shrink a file system, which is beyond the scope of this document.

## Step 13 Reboot!

Reset and Reboot the Beagle Bone Black.

## Step 14 Fix the "fsck" problem (see above)

Once the BBB has rebooted, login.

We now solve the 'fsck' error we ignored above. The details are discussed in this Debian Form posting:

<http://forums.debian.net/viewtopic.php?f=17&t=127039>

The solution is to run this command:

**debian@beaglebone:~$ sudo update-initramfs -u**

**(Enter password if needed)**

**update-initramfs: Generating /boot/initrd.img-4.4.54-ti-r93**

## The End