Speicher

Partition		SD card	eMMC	Update(BPI-Tool)	Update(RAW command)
BootLoader	Header	0-2KB	Boot0: 0-2KB	SD (On HOST PC or target board) : *bpi-bootsel BPI-R2-720P-2k.img.gz /dev/mmcblkX* (for updating both uboot and preloader) eMMC(Only on target board): *echo 0 > /sys/block/mmcblkYboot0/force_ro* *bpi-bootsel BPI-R2-EMMC-boot0-DDR1600-0k-0905.img.gz /dev/mmcblkyboot0* (for updating bootloader to EMMC boot0) *bpi-bootsel BPI-R2-720P-2k.img.gz /dev/mmcblkY* (for updating Uboot to eMMC user block)	We just have RAW command to update EMMC preloader under uboot: *emmc write 1 84000000 0 200* Write Uboot to SD or EMMC under uboot: mmc dev Z mmc write 84000000 280 200 Where Z is the number of disk, 0-emmc, 1-SD
	Preloader	2-320KB	Boot0: 0-320KB		
	Uboot	320KB - 1MB	User Data Area: 320KB - 1MB		
	Reserved	1MB - 100MB	User Data Area: 1MB - 100MB		
File System Partition1(FAT32)		100MB - 356MB	User Data Area:100MB - 356MB		
File System Partition2(EXT4)		356MB - 7456MB	User Data Area:356MB - 7456MB	cd SD;bpi-update -c bpi-r2.conf -d /dev/<device>	

Quelle: http://forum.banana-pi.org/t/how-to-update-uboot-without-bpi-update/4023/2

es sieht so aus, als wenn /dev/mmcblk1, /dev/mmcblk1boot0 und /dev/mmcblk1boot1 unabhängige Geräte sind (bootx nicht Partitionen in /dev/mmcblk1)

preloader

for SD

muss am 2k-offset (0x800) geschrieben werden

```
sudo dd if=BPI-R2-EMMC-boot0-DDR1600-20190722-2k.img of=/dev/sdc bs=1k seek=2
```

(SD-Card benötigt zusätzlich MMC_BOOT & BRLYT header, siehe weiter unten)

for EMMC

muss am 0-offset (0x0) der boot0-Partition geschrieben werden

```
sudo dd if=BPI-R2-EMMC-boot0-DDR1600-20190722-0k.img of=/dev/mmcblk1boot0
```

Dateien von hier: https://github.com/BPI-SINOVOIP/BPI-files/tree/master/SD/100MB

SD

sdcard-bootsektor reverse-engineering
http://forum.banana-pi.org/t/boot-fails-with-self-build-u-boot/5460/20

http://forum.banana-pi.org/t/how-to-build-an-ubuntu-debian-sd-image-from-scratch/6805/8

SD-Headers

bpi-r2-head440-0k.img
bpi-r2-head1-512b.img

- SDMMC_BOOT-Signatur + Adresse des 2. Headers (0x00000200) - erste 440 byte (vor Partitionstabelle):

```
gunzip -c BPI-R2-HEAD440-0k.img.gz | dd of=/dev/loop8 bs=1024 seek=0
```

- BRLYT-Signatur + Preloader-Adresse (0x00000800):

```
gunzip -c BPI-R2-HEAD1-512b.img.gz | dd of=/dev/loop8 bs=512 seek=1
```

komplett

```
dd if=/dev/zero of=../bpi-r2-buster.img bs=1M count=7168
loopdev=$(losetup -f)
sudo losetup ${loopdev} ../bpi-r2-buster.img
echo $loopdev
sudo dd if=~/Downloads/BPI-R2-preloader-DDR1600-20190722-2k.img
of=${loopdev} bs=1k seek=2
sudo dd if=~/Downloads/BPI-R2-HEAD440-0k.img of=${loopdev} bs=1024 seek=0
sudo dd if=~/Downloads/BPI-R2-HEAD1-512b.img  bs=512 seek=1
sudo dd if=/path/to/u-boot/u-boot.bin of=${loopdev} bs=1k seek=320
sudo sfdisk ${loopdev} < ~/Downloads/parttable.dat
sudo partprobe ${loopdev}
ls ${loopdev}*
sudo mkfs -t vfat ${loopdev}p1
sudo mkfs -t ext4 ${loopdev}p2
sudo fatlabel ${loopdev}p1 BPI-BOOT
sudo e2label ${loopdev}p2 BPI-ROOT
```

install debian (from bootstrapped rootfs)

```
sudo mount ${loopdev}p2 /mnt/
sudo cp -r debian_buster_armhf/. /mnt/
#install kernel-modules to same partition
kernelpack=/path/to/bpi-r2_<version>_main.tar.gz
sudo tar -xzf ${kernelpack} -C /mnt/ --strip-components=1 BPI-ROOT
#install kernel to boot-partition
sudo umount /mnt
sudo mount ${loopdev}p1 /mnt/
sudo tar -xzf ${kernelpack} -C /mnt/ --strip-components=1 BPI-BOOT
#maybe create a uEnv.txt
sudo umount /mnt
```

```
sudo losetup -d ${loopdev}
#now write the image to card (make sure /dev/sdc is your sdcard-device and
no partition is mounted)
sudo dd if=../bpi-r2-buster.img of=/dev/sdc
sync
```

MMC-Utils

über die mmc-utils kann man aus einem laufenden System testen, ob die EMMC-Partitionierung stimmt (sollte 0x48 sein siehe partition-konfiguration_des_emmc_aendern).

```
./mmc extcsd read /dev/mmcblk1
....
Boot configuration bytes [PARTITION_CONFIG: 0x48]
....
```

ich habe die mmc-utils auch in mein Kernel-Repo übernommen (mit angepasstem Makefile für Cross-Compile)

laut einem Forum-Nutzer (siehe hier) lässt sich die partition config mit den mmc-utils auch schreiben

```
./mmc bootpart enable 1 1 /dev/mmcblk1
```

```
[18:02] root@bpi-r2:~# ./mmc extcsd read /dev/mmcblk1 | grep
PARTITION_CONFIG
Boot configuration bytes [PARTITION_CONFIG: 0x00]
[18:02] root@bpi-r2:~# ./mmc bootpart enable 1 1 /dev/mmcblk1
[18:03] root@bpi-r2:~# ./mmc extcsd read /dev/mmcblk1 | grep
PARTITION_CONFIG
Boot configuration bytes [PARTITION_CONFIG: 0x48]
```

Betriebssystem auf EMMC installieren

http://forum.banana-pi.org/t/bpi-r2-new-image-ubuntu-16-04-v1-2-1-bt-and-wifi-ap-mode-are-working-fine-2017-11-27/4291

1. partition-konfiguration_des_emmc_aendern
2. Schreibmodus für /dev/mmcblk1boot0 aktivieren:

   ```
   echo 0 > /sys/block/mmcblk1boot0/force_ro
   ```

3. Preloader von hier auf das boot-device schreiben:

   - ```
     gunzip -c BPI-R2-EMMC-boot0-DDR1600-0k-0905.img.gz | sudo dd
     of=/dev/mmcblk1boot0 bs=1024 seek=0
     ```

   - mit bpi-tools:

     ```
     bpi-bootsel BPI-R2-EMMC-boot0-DDR1600-0k-0905.img.gz
     /dev/mmcblk1boot0
     ```

4. kopieren des OS-Abbildes auf EMMC (device=/dev/mmcblk1):

   - ```
     unzip -p <XXX.img.zip> | pv | dd of=<device> bs=10M status=noxfer
     ```

- Alternative (mit bpi-tools):

```
bpi-copy  <XXX.img.zip> <device>
```

5. Ausschalten, SD entfernen und neu hochfahren

wenn sd-Karten-Abbild nicht auf emmc passt: Abbild verkleinern

manuelles kopieren des Betriebssystems

- für ein neues SD-Card-Image wird der Bootblock eines vorhandenen Images benötigt
  - erste 2k ohne preloader/uboot
    erstes MB mit preloader/uboot

```
gunzip bpi-r2-sd-boot*.img.gz
dd if=bpi-r2-sd-boot1m.img of=/dev/sdx
#ggf. partitonstabelle neu einlesen:
sfdisk -R /dev/sdx
#alternativ aus Paket parted
partprobe /dev/sdx
```

- uboot installieren:

```
dd if=BPI-R2-720P-2k.img of=/dev/mmcblk1 bs=1k seek=2 count=1022
```

- Partitionstabelle auf der SD exportieren
  parttable.dat
  und auf emmc einspielen:

```
root@bpi-r2:~# sfdisk -d /dev/mmcblk0 > parttable.dat
root@bpi-r2:~# sfdisk /dev/mmcblk1 < parttable.dat
```

- ggf. checken/vergrößern
- Dateisysteme anlegen (mkfs) für p1=vfat (apt-get install dosfstools) und p2=ext4

```
mkfs -t vfat /dev/mmcblk1p1
mkfs -t ext4 /dev/mmcblk1p2
```

- im bestehenden System mount-Punkte anlegen/konfigurieren

```
mkdir -p /mnt/emmc/boot
mkdir -p /mnt/emmc/root
nano /etc/fstab
# <file system>         <dir>           <type>  <options>
<dump>  <pass>
/dev/mmcblk0p2          /               ext4    errors=remount-ro
0       1
/dev/mmcblk0p1          /boot           vfat    defaults
0       0
/dev/mmcblk1p2          /mnt/emmc/root  ext4    errors=remount-
ro,noauto   0       1
```

```
/dev/mmcblk1p1          /mnt/emmc/boot  vfat    defaults,noauto
0       0
```

- mounten:

```
mount /mnt/emmc/root
mount /mnt/emmc/boot
```

- rootfs entpacken/rüberkopieren

```
rsync -aAXv --
exclude={"/dev/*","/proc/*","/sys/*","/tmp/*","/run/*","/mnt/*","/media
/*","/lost+found","/boot/*"} / /mnt/emmc/root/
```

- kernel (p1) und Module (p2) rüberkopieren

```
mkdir -p /mnt/emmc/boot/bananapi/bpi-r2/linux
cp /boot/bananapi/bpi-r2/linux/uImage /mnt/emmc/boot/bananapi/bpi-
r2/linux
mkdir -p /mnt/emmc/root/lib/modules/
cp -r /lib/modules/$(uname -r) /mnt/emmc/root/lib/modules/
```

- uboot auf die richtige Partition konfigurieren

```
sed 's/mmcblk0/mmcblk1/' /boot/bananapi/bpi-r2/linux/uEnv.txt >
/mnt/emmc/boot/bananapi/bpi-r2/linux/uEnv.txt
```