UBoot

erreichbar über debug-uart

```
    *** U-Boot Boot Menu ***
       1. System Load Linux to SDRAM via TFTP.
       2. System Load Linux Kernel then write to Flash via TFTP.
       3. Boot Linux from SD.
       4. System Load Boot Loader then write to Flash via TFTP.
       5. System Load Linux Kernel then write to Flash via Serial.
       6. System Load Boot Loader then write to Flash via Serial.
       7. Boot system code via Flash.
       U-Boot console       <<<<<<<<
    Press UP/DOWN to move, ENTER to select
```

Uboot erneuern

Der emmc-Befehl ist erst seit dem 29.September 2017 im uboot (Version: „U-Boot 2014.04-rc1 (Oct 16 2017 - 19:33:23)“)

U-Boot von GitHub kompilieren

```
sudo dd if=/dev/sdx of=bpi-r2-first10M.img bs=1M count=10 #Backup der ersten 10MB

SD/100MB$ gunzip BPI-R2-720P-2k.img.gz
SD/100MB$ sudo dd if=BPI-R2-720P-2k.img of=/dev/sdx bs=1k seek=2 count=1022 #unzipped img!

sudo dd of=/dev/sdx if=bpi-r2-first10M.img bs=1M count=10 #die ersten 10MB wiederherstellen (bei Fehler)
```

alternativ nur uboot (nach option 2 in build.sh, backup nicht vergessen):

```
sudo dd of=/dev/sdb if=u-boot-mt/u-boot.bin bs=1k seek=320
```

vorcompiliertes uboot-image kann auf meinem gDrive herunter geladen werden. oder die bin-Datei

hier

Quelle für Position des BPI-R2-720p-Images:
https://github.com/BPI-SINOVOIP/bpi-tools/blob/beb36af51a4b455a2a09ec9348a6efca1fe390cc/bpi-bootsel#L245

Zusammensetzung des Images:
https://github.com/BPI-SINOVOIP/BPI-R2-bsp/blob/d94f55022a9192cb181d380b1a6699949a36f30c/scripts/bootloader.sh#L19

```
TMP_FILE=${U}/${BOARD}.tmp
IMG_FILE=${U}/${BOARD}-2k.img
PRELOADER=$TOPDIR/mt-
pack/mtk/${TARGET_PRODUCT}/bin/preloader_iotg7623Np1_emmc.bin
UBOOT=$TOPDIR/u-boot-mt/u-boot.bin

(sudo dd if=$PRELOADER  of=${LOOP_DEV} bs=1k seek=2) >/dev/null 2>&1
(sudo dd if=$UBOOT  of=${LOOP_DEV} bs=1k seek=320) >/dev/null 2>&1
(dd if=${TMP_FILE} of=${IMG_FILE} bs=1k skip=2 count=1022 status=noxfer)
>/dev/null 2>&1
```

es wird also die compilierte uboot.bin (u-boot-mt/u-boot.bin) verwendet und diese liegt auf der SD-Karte an position 0x50000 (320k), der Preloader (mt-pack/mtk/bpi-r2/bin/preloader_iotg7623Np1_emmc.bin) liegt an position 0x800 (2k) der SD-Karte

uboot 2018-11

Mediatek hat patches für den BPI-R2 geposted…diese habe ich einem uboot-fork angewendet und eingerichtet (build.sh, config, default-Environment, …): https://github.com/frank-w/u-boot

Kernel von der SD-Karte lassen sich bereits starten (emmc sollte auch funktionieren), mittlerweile gibt es auch einen Ethernet-Treiber

falls nicht das default environment geladen wurde (buildargs):

```
env default -a
printenv
#saveenv
```

Liste der Befehle

zurück zum Menü mit dem Befehl „bootmenu“

```
BPI-IoT> help
```

```
?       - alias for 'help'
backup_message- print backup message.
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootm   - boot application image from memory
bootmenu- ANSI terminal bootmenu
bootp   - boot image via network using BOOTP/TFTP protocol
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
```

```
crc32   - checksum calculation
echo    - echo args to console
editenv - edit environment variable
emmc    - eMMC sub system
env     - environment handling commands
esw_read- esw_read   - Dump external switch/GMAC status !!

exit    - exit script
false   - do nothing, unsuccessfully
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
go      - start application at address 'addr'
help    - print command description/usage
image_blks- read image size from img_size or image header if no specifying
img_.
image_check- check if image in load_addr is normal.
iminfo  - print header information for application image
imxtract- extract a part of a multi-image
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loadx   - load binary file over serial line (xmodem mode)
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
md      - memory display
mdio    - mdio   - Ralink PHY register R/W command !!

mm      - memory modify (auto-incrementing address)
mmc     - MMC sub-system
mmc2    - MMC sub system
mmcinfo - display MMC info
mtk_image_blks- read image size from image header (MTK format) located at
load_.
mw      - memory write (fill)
nm      - memory modify (constant address)
nor     - nor   - nor flash command

ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reco_message- print recovery message.
reg     - reg   - Ralink PHY register R/W command !!

reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
serious_image_check- seriously check if image in load_addr is normal.
setenv  - set environment variables
showvar - print local hushshell variables
sleep   - delay execution for some time
snor    - snor   - spi-nor flash command
```

```
source  - run script from memory
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
true    - do nothing, successfully
uboot_check- check if uboot in load_addr is normal.
version - print monitor, compiler and linker version
```

Partition-Konfiguration des EMMC ändern

```
BPI-IoT> emmc --help
emmc - eMMC sub system
Usage:
emmc read part addr blk# cnt
emmc write part addr blk# cnt
emmc ecsd       - Dump ext csd
emmc pconf val  - Set Part Config val
BPI-IoT> emmc ecsd
```

emmc ecsd

```
============================================================
[EXT_CSD] EXT_CSD rev.              : v1.7 (MMCv5.0)
[EXT_CSD] CSD struct rev.           : v1.2
[EXT_CSD] Supported command sets    : 1h
[EXT_CSD] HPI features              : 1h
[EXT_CSD] BG operations support     : 1h
[EXT_CSD] BG operations status      : 0h
[EXT_CSD] Correct prg. sectors      : 0h
[EXT_CSD] 1st init time after part. : 3000 ms
[EXT_CSD] Min. write perf.(DDR,52MH,8b): 0h
[EXT_CSD] Min. read perf. (DDR,52MH,8b): 0h
[EXT_CSD] TRIM timeout: 0 ms
[EXT_CSD] Secure feature support: 55h
[EXT_CSD] Secure erase timeout  : 8100 ms
[EXT_CSD] Secure trim timeout   : 5100 ms
[EXT_CSD] Access size           : 3072 bytes
[EXT_CSD] HC erase unit size    : 512 kbytes
[EXT_CSD] HC erase timeout      : 300 ms
[EXT_CSD] HC write prot grp size: 8192 kbytes
[EXT_CSD] HC erase grp def.     : 0h
[EXT_CSD] Reliable write sect count: 1h
[EXT_CSD] Sleep current (VCC) : 7h
[EXT_CSD] Sleep current (VCCQ): 7h
[EXT_CSD] Sleep/awake timeout : 26214400 ns
[EXT_CSD] Sector count : e90000h
[EXT_CSD] Min. WR Perf.  (52MH,8b): 0h
[EXT_CSD] Min. Read Perf.(52MH,8b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,8b,52MH,4b): 0h
```

```
[EXT_CSD] Min. Read Perf.(26MH,8b,52MH,4b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,4b): 0h
[EXT_CSD] Min. Read Perf.(26MH,4b): 0h
[EXT_CSD] Power class: 0
[EXT_CSD] Power class(DDR,52MH,3.6V): 0h
[EXT_CSD] Power class(DDR,52MH,1.9V): 0h
[EXT_CSD] Power class(26MH,3.6V)    : 0h
[EXT_CSD] Power class(52MH,3.6V)    : 0h
[EXT_CSD] Power class(26MH,1.9V)    : 0h
[EXT_CSD] Power class(52MH,1.9V)    : 0h
[EXT_CSD] Part. switch timing    : 1h
[EXT_CSD] Out-of-INTR busy timing: 5h
[EXT_CSD] Card type       : 57h
[EXT_CSD] Command set     : 0h
[EXT_CSD] Command set rev.: 0h
[EXT_CSD] HS timing       : 0h
[EXT_CSD] Bus width       : 0h
[EXT_CSD] Erase memory content : 0h
[EXT_CSD] Partition config      : 0h      <<<<<<<<<<<<<<<<<<<<<<<<<<<<
falsche Partitionskonfiguration
[EXT_CSD] Boot partition size   : 4096 kbytes
[EXT_CSD] Boot information      : 7h
[EXT_CSD] Boot config protection: 0h
[EXT_CSD] Boot bus width        : 0h
[EXT_CSD] Boot area write prot  : 0h
[EXT_CSD] User area write prot  : 0h
[EXT_CSD] FW configuration      : 0h
[EXT_CSD] RPMB size : 512 kbytes
[EXT_CSD] Write rel. setting  : 1fh
[EXT_CSD] Write rel. parameter: 4h
[EXT_CSD] Start background ops : 0h
[EXT_CSD] Enable background ops: 0h
[EXT_CSD] H/W reset function   : 0h
[EXT_CSD] HPI management       : 0h
[EXT_CSD] Max. enhanced area size : 136h (2539520 kbytes)
[EXT_CSD] Part. support  : 7h
[EXT_CSD] Part. attribute: 0h
[EXT_CSD] Part. setting  : 0h
[EXT_CSD] General purpose 1 size : 0h (0 kbytes)
[EXT_CSD] General purpose 2 size : 0h (0 kbytes)
[EXT_CSD] General purpose 3 size : 0h (0 kbytes)
[EXT_CSD] General purpose 4 size : 0h (0 kbytes)
[EXT_CSD] Enh. user area size : 0h (0 kbytes)
[EXT_CSD] Enh. user area start: 0h
[EXT_CSD] Bad block mgmt mode: 0h
```

```
  ============================================================
  BPI-IoT> emmc pconf 0x48      <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
partition config ändern
  BPI-IoT> emmc ecsd            <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
verifizieren
  ============================================================
```

emmc ecsd (nachher)

```
[EXT_CSD] EXT_CSD rev.              : v1.7 (MMCv5.0)
[EXT_CSD] CSD struct rev.           : v1.2
[EXT_CSD] Supported command sets    : 1h
[EXT_CSD] HPI features              : 1h
[EXT_CSD] BG operations support     : 1h
[EXT_CSD] BG operations status      : 0h
[EXT_CSD] Correct prg. sectors      : 0h
[EXT_CSD] 1st init time after part. : 3000 ms
[EXT_CSD] Min. write perf.(DDR,52MH,8b): 0h
[EXT_CSD] Min. read perf. (DDR,52MH,8b): 0h
[EXT_CSD] TRIM timeout: 0 ms
[EXT_CSD] Secure feature support: 55h
[EXT_CSD] Secure erase timeout  : 8100 ms
[EXT_CSD] Secure trim timeout   : 5100 ms
[EXT_CSD] Access size           : 3072 bytes
[EXT_CSD] HC erase unit size    : 512 kbytes
[EXT_CSD] HC erase timeout      : 300 ms
[EXT_CSD] HC write prot grp size: 8192 kbytes
[EXT_CSD] HC erase grp def.     : 0h
[EXT_CSD] Reliable write sect count: 1h
[EXT_CSD] Sleep current (VCC) : 7h
[EXT_CSD] Sleep current (VCCQ): 7h
[EXT_CSD] Sleep/awake timeout : 26214400 ns
[EXT_CSD] Sector count : e90000h
[EXT_CSD] Min. WR Perf.  (52MH,8b): 0h
[EXT_CSD] Min. Read Perf.(52MH,8b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,8b,52MH,4b): 0h
[EXT_CSD] Min. Read Perf.(26MH,8b,52MH,4b): 0h
[EXT_CSD] Min. WR Perf.  (26MH,4b): 0h
[EXT_CSD] Min. Read Perf.(26MH,4b): 0h
[EXT_CSD] Power class: 0
[EXT_CSD] Power class(DDR,52MH,3.6V): 0h
[EXT_CSD] Power class(DDR,52MH,1.9V): 0h
[EXT_CSD] Power class(26MH,3.6V)    : 0h
[EXT_CSD] Power class(52MH,3.6V)    : 0h
[EXT_CSD] Power class(26MH,1.9V)    : 0h
[EXT_CSD] Power class(52MH,1.9V)    : 0h
[EXT_CSD] Part. switch timing    : 1h
[EXT_CSD] Out-of-INTR busy timing: 5h
[EXT_CSD] Card type       : 57h
[EXT_CSD] Command set     : 0h
[EXT_CSD] Command set rev.: 0h
[EXT_CSD] HS timing       : 1h
[EXT_CSD] Bus width       : 0h
[EXT_CSD] Erase memory content : 0h
```

```
[EXT_CSD] Partition config      : 48h     <<<<<<<<<<<<<<<<<<<<<<<<<<<<
[EXT_CSD] Boot partition size   : 4096 kbytes
[EXT_CSD] Boot information      : 7h
[EXT_CSD] Boot config protection: 0h
[EXT_CSD] Boot bus width        : 0h
[EXT_CSD] Boot area write prot  : 0h
[EXT_CSD] User area write prot  : 0h
[EXT_CSD] FW configuration      : 0h
[EXT_CSD] RPMB size : 512 kbytes
[EXT_CSD] Write rel. setting  : 1fh
[EXT_CSD] Write rel. parameter: 4h
[EXT_CSD] Start background ops : 0h
[EXT_CSD] Enable background ops: 0h
[EXT_CSD] H/W reset function   : 0h
[EXT_CSD] HPI management       : 0h
[EXT_CSD] Max. enhanced area size : 136h (2539520 kbytes)
[EXT_CSD] Part. support  : 7h
[EXT_CSD] Part. attribute: 0h
[EXT_CSD] Part. setting  : 0h
[EXT_CSD] General purpose 1 size : 0h (0 kbytes)
[EXT_CSD] General purpose 2 size : 0h (0 kbytes)
[EXT_CSD] General purpose 3 size : 0h (0 kbytes)
[EXT_CSD] General purpose 4 size : 0h (0 kbytes)
[EXT_CSD] Enh. user area size : 0h (0 kbytes)
[EXT_CSD] Enh. user area start: 0h
[EXT_CSD] Bad block mgmt mode: 0h
============================================================
```

in neueren uboot-Versionen (2018):

http://forum.banana-pi.org/t/add-latest-u-boot-support-for-bpi-r2-bpi-r64-not-yet/6938/26

```
mmc partconf 0 1 1 0
```

System von Console starten

```
BPI-IoT> printenv
...
boot10=mmc init; run boot_normal; bootm
...
bootmenu_2=3. Boot Linux from SD.=run boot10
...
```

```
run boot10
```

Kernel angeben

in der BPI-BOOT/bananapi/bpi-r2/linux/uEnv.txt den parameter kernel anpassen:

```
#kernel=uImage
#kernel=uImage_4.14.33
kernel=uImage_4.9.92
```

dies hat den Vorteil, dass man einen neuen Kernel testen kann und notfalls auf den alten leicht wieder zurück kann (wenn diese 2 verschiedene Namen haben). Für Multiboot muss der uboot-code angepasst werden, da die uEnv.txt erst mit dem Menüpunkt „Boot Linux from SD“ geladen wird…vorher sieht man seine eigenen Variablen nicht.

uEnv.txt laden

Standardmäßig wird die uEnv.txt erst geladen wenn der Menüpunkt „Boot from SD“ ausgewählt wurde.

```
#Boot from emmc
enter to uboot-console
execute “mmc init 0”
execute “setenv partition 0:1”
execute “run loadbootenv”
execute “env import -t ${scriptaddr} ${filesize} ”

#Boot from SD:
enter to uboot-console
execute “mmc init 1”
execute “setenv partition 1:1”
execute “run loadbootenv”
execute “env import -t ${scriptaddr} ${filesize} ”
```

Quelle: http://forum.banana-pi.org/t/how-to-extend-the-uboot-menu/5415/7

da loadbootenv eine Variable ist, die nur im offiziellen bpi-r2-uboot definiert ist und u.a. im U-Boot-Upstream-repo nicht existiert müssen folgende Variablen definiert werden um die uEnv.txt + kernel zu laden

```
setenv scriptaddr 0x83000000
setenv bpi bananapi
setenv board bpi-r2
setenv service linux
setenv device mmc
setenv partition 1:1
setenv bootenv uEnv.txt
setenv loadbootenv fatload ${device} ${partition} ${scriptaddr}
${bpi}/${board}/${service}/${bootenv}
setenv importenv env import -t ${scriptaddr} ${filesize}
```

```
run loadbootenv
run importenv

printenv

setenv newboot "fatload ${device} ${partition} ${loadaddr}
${bpi}/${board}/${service}/${kernel}; bootm"
run newboot

#check for boot-device (emmc/sd)
setenv checksd fatinfo ${device} 1:1
setenv selectmmc "if run checksd; then echo Boot from SD ; setenv partition
1:1;else echo Boot from eMMC; setenv partition 0:1 ; fi;"

run selectmmc
```

nützliche Befehle

MMC

```
U-Boot> mmc list
mmc@11230000: 0 (eMMC)
mmc@11240000: 1 (SD)

#set mmc-device (1=sd,0=emmc)
U-Boot> mmc dev 1

#read current device
U-Boot> mmc dev
switch to partitions #0, OK
mmc1 is current device

U-Boot> mmcinfo
Device: mmc@11240000
Manufacturer ID: 1b
OEM: 534d
Name: 00000
Bus Speed: 50000000
Mode : SD High Speed (50MHz)
Rd Block Len: 512
SD version 2.0
High Capacity: Yes
Capacity: 7.6 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes

sd-card (1) has 4-bit bus width, emmc (0) has 8-bit bus width
```

```
#partitionconfig
#mmc partconf dev [boot_ack boot_partition partition_access]
# - Show or change the bits of the PARTITION_CONFIG field of the specified
device
#example for mode 0x48 (needed for emmc-boot on bpi-r2)
U-Boot> mmc partconf 0
EXT_CSD[179], PARTITION_CONFIG:
BOOT_ACK: 0x1
BOOT_PARTITION_ENABLE: 0x1
PARTITION_ACCESS: 0x0

#setzen mit
U-Boot> mmc partconf 0 1 1 0
```

environment löschen/neu schreiben

```
#zur Sicherheit kann man sich das geschriebene env anschauen (vorher auf sd
wechseln,wenn davon gebootet):
BPI-R2> mmc dev 1
BPI-R2> mmc read ${scriptaddr} 800 10

MMC read: dev # 1, block # 2048, count 16 ... 16 blocks read: OK
BPI-R2> strings ${scriptaddr}
p��saskkernel=askenv kernelinput "enter uImage-name:";
...
#die ersten 4 bytes sind die CRC-Prüfsumme, danach geht das env los...

#environment löschen (ab block #800h=1MB/512b 16 Blöcke á 512b=8k => 10h)
verwendet eigenen mmc erase befehl
#BPI-R2> mmc erase 800 10
#wird mit eraseenv (patchwork) abgelöst
eraseenv

#default-environment laden
BPI-R2> env default -a;

#environment neu schreiben
BPI-R2> saveenv
Saving Environment to MMC... Boot From SD(id:1)

Writing to MMC(1)... OK
```

mehr zu den mmc-Kommandos hier

Verzeichnisauflistung

```
ls mmc 1:1 bananapi/bpi-r2/linux
#mit den Variablen aus meiner U-boot-Umgebung:
```

```
ls ${device} ${partition} ${bpi}/${board}/${service}
```

kernelabfrage

```
lskernel=ls ${device} ${partition} ${bpi}/${board}/${service};
askkernel=askenv kernelinput "enter uImage-name:";
boot0=run lskernel;run askkernel;if printenv kernelinput ;then setenv kernel
${kernelinput}; run newboot; fi
bootmenu_0=1. Enter kernel-name to boot from SD/EMMC.=run boot0
```

prüfen, ob datei existiert

```
checkenv=test -e ${device} ${partition}
${bpi}/${board}/${service}/${bootenv}
#will be evaluated to check if bananapi/bpi-r2/linux/uEnv.txt (device=mmc,
partition=1:1 for sdcard)
if run checkenv; then ...; else echo file not found; fi;
```

anderes uboot via TFTP nachladen

(benötigt CONFIG_CMD_CACHE)

```
BPI-R2> tftp 0x81E00000 ${serverip}:u-boot_2019.07-rc4-bpi-r2-dbg.bin
BPI-R2> icache off;dcache off
BPI-R2> go 0x81E00000
```

PCIe


uboot vor 2020-10 (meine version) hat einen Bug welcher beim „pci enum“ hängen bleibt, wenn keine Karte im pcie-slot gesteckt ist

```
BPI-R2> pci enum
BPI-R2> pci 0
Scanning PCI devices on bus 0
BusDevFun  VendorId   DeviceId   Device Class       Sub-Class
_____________________________________________________________
00.00.00   0x14c3     0x0801     Bridge device           0x04
00.01.00   0x14c3     0x0801     Bridge device           0x04
BPI-R2> pci 1
Scanning PCI devices on bus 1
BusDevFun  VendorId   DeviceId   Device Class       Sub-Class
_____________________________________________________________
01.00.00   0x14c3     0x7612     Network controller      0x80
BPI-R2> pci 2
```

```
Scanning PCI devices on bus 2
BusDevFun  VendorId   DeviceId   Device Class       Sub-Class
_____________________________________________________________
02.00.00   0x1b21     0x0611     Mass storage controller 0x01
BPI-R2> scsi scan
scanning bus for devices...
SATA link 0 timeout.
Target spinup took 0 ms.
AHCI 0001.0200 32 slots 2 ports 6 Gbps 0x3 impl SATA mode
flags: 64bit ncq stag led clo pmp pio slum part ccc sxs
  Device 0: (1:0) Vendor: ATA Prod.: ST750LM022 HN-M7 Rev: 2AR1
            Type: Hard Disk
            Capacity: 715404.8 MB = 698.6 GB (1465149168 x 512)
BPI-R2>
```

SATA

siehe pcie (pci enum + scsi scan) und dann via

```
ls scsi 0:1
```

auf die HDD zugreifen

USB

```
BPI-R2> usb start
starting USB...
Bus usb@1a1c0000: hcd: 0x1a1c0000, ippc: 0x1a1c4700
u2p:1, u3p:1
Register 200010f NbrPorts 2
Starting the controller
USB XHCI 0.96
Bus usb@1a240000: hcd: 0x1a240000, ippc: 0x1a244700
u2p:1, u3p:1
Register 200010f NbrPorts 2
Starting the controller
USB XHCI 0.96
scanning bus usb@1a1c0000 for devices... 1 USB Device(s) found
scanning bus usb@1a240000 for devices... 2 USB Device(s) found
       scanning usb for storage devices... 1 Storage Device(s) found
BPI-R2> usb tree
USB device tree:
  1  Hub (5 Gb/s, 0mA)
     U-Boot XHCI Host Controller

  1  Hub (5 Gb/s, 0mA)
  |  U-Boot XHCI Host Controller
  |
```

```
  +-2  Mass Storage (480 Mb/s, 200mA)
       USB      Flash Disk       906B030002F4

BPI-R2> ls usb 0:1
            efi/
  4767728   kernel

1 file(s), 1 dir(s)
```

dt overlays

https://forum.banana-pi.org/t/set-mac-address-on-boot/7224/4

Overlay muss mit -@ compiliert werden

```
dtc -@ -I dts -O dtb -o bpi-r2-mac.dtb bpi-r2-mac.dts
```

Sonst kommt die Meldung beim laden:

```
failed on fdt_overlay_apply(): FDT_ERR_NOTFOUND base fdt does did not have a
/symbols node make sure you’ve compiled with -@
```

Die Haupt-DTB muss auch mit -@ compiliert werden...dazu kann man beim lernel kompilieren die Variable DTC_FLAGS setzen:

```
export DTC_FLAGS=-@
```

Verfizieren lässt sich das mit fdtdump

```
fdtdump arch/.../boot/dts/.../board.dtb | grep -C3 __symbols__
```

Hier sollten die Namen der dts nodes auftauchen

Beim laden in uboot muss erst die haupt-dtb geladen werden und dann das overlay. Dafür habe ich in meinem uboot folgende Variablen definiert (ausführen via run $varname)

```
loadfdt=fatload ${device} ${partition} ${dtaddr}
${bpi}/${board}/${service}/dtb/${fdt}
loaddto=echo "loaddto:${dto}";fdt addr ${dtaddr};fdt resize 8192; setexpr
fdtovaddr ${dtaddr} + F000;fatload ${device} ${partition} ${fdtovaddr}
${bpi}/${board}/${service}/dtb/${dto} && fdt apply ${fdtovaddr}
```

Links

Patchwork Archiv

git