Hack the Gadget!

Daniel Maslowski
Agenda

Hands-on Hardware Hacking
Bringing up your device
Hands-on Hardware Hacking
Things I hack on

- mediaplayers
- TVboxes
- NVRs/DVRs
- cameras
- routers
- wireless storages
- laptops, desktops, SBCs
Things I hack on

- media players
- TV boxes
- NVRs / DVRs
- cameras
- routers
- wireless storages
- laptops, desktops, SBCs
PCB mess
PCB mess
PCB mess

WHERE IS THE UART?!
PCB mess resolved

- Checked pins with multimeter
  - Measured voltage
    - All voltages max 3V
  - Attach USB serial RX
  - Got nothing, now what?
  - Read about other products
  - OpenWrt forum rocks
  - Set the baud rate to 57600
    - Got output -> TX
  - RX is likely nearby
    - Poked around -> bingo!
  - Typing echo back
PCB mess resolved

- checked pins with multimeter
  - measured voltage

- all voltages max 3V
- attach USB serial RX
- got nothing, now what?
- read about other products
- OpenWrt forum rocks
- set the baud rate to 57600
- got output -> TX
- RX is likely nearby
- poked around -> bingo!
- typing echo backed
PCB mess resolved

- checked pins with multimeter
  - measured voltage
  - all voltages max 3V
  - attach USB serial RX
  - got nothing, no what?

- OpenWrt forum rocks
  - set the baud rate to 57600
  - got output -> TX
  - RX is likely nearby
  - poked around -> bingo!
  - typing echo back
PCB mess resolved

- checked pins with multimeter
  - measured voltage
  - all voltages max 3V
  - attach USB serial RX
  - got nothing, no what?
- read about other products
  - OpenWrt forum rocks
PCB mess resolved

- checked pins with multimeter
  - measured voltage
  - all voltages max 3V
  - attach USB serial RX
  - got nothing, no what?
- read about other products
  - OpenWrt forum rocks
- set the baud rate to 57600
  - got output -> TX

```
#  Note
```
PCB mess resolved

- checked pins with multimeter
  - measured voltage
  - all voltages max 3V
- attach USB serial RX
- got nothing, no what?
- read about other products
  - OpenWrt forum rocks
  - set the baud rate to 57600
  - got output -> TX

RX is likely nearby
- poked around -> bingo!
- typing echoed back
Some of those details are lies: the F133 (Allwinner SoC aka D1s) only has 512 Mbit DDR2 DRAM in-package, or 64 MiB. Is 1024x600 really HD?...
Car Media Player

Product details
- CPU F133
- 1 Gbit DRAM memory.
- Memory None
- 1024 x 600 screen resolution.
- 7 inch HD screen size

Portable Wireless Carplay Android Car Stereo 7 Inch HD Touchscreen Car MP5 Player with Mirrorlink Remote Control FM Radio USB 12 LED Camera

Brand: wempecker

Prices for items sold by Amazon include VAT. Depending on your delivery address, VAT may vary at Checkout. For other items, please see details.

Brand: wempecker
Connectivity technology: Bluetooth, Auxiliary, Wi-Fi, USB
Controller type: Android
Compatible devices: Smartphone, Speaker
Connector Type: USB Type A, 3.5 mm Klinkke
Audio output mode: Stereo
Control method: Touch

About this item
- Mirror Link: This full touch screen car radio supports Mirror Link for iOS and Android smartphones. You can sync maps, movies etc. on the large 7 inch screen. The full touch HD display with a resolution of 1024 x 600 provides you with a clear and responsive viewing experience. Equipped with a remote control, it offers you a more convenient experience.
Car Media Player

Portable Wireless Carplay Android Car Stereo 7 Inch HD Touchscreen Car MP5 Player with Mirrorlink Remote Control FM Radio USB 12 LED Camera

Brand: vepeculer

CPU: F133

1 Gbit DRAM memory.

Memory: None

1024 x 600 screen resolution.

7 inch HD screen size

Some of those details are lies: the F133 (Allwinner SoC aka D1s) only has 512 Mbit DDR2 DRAM in-package, or 64MiB. Is 1024 x 600 really HD? …

https://www.amazon.de/-/en/Portable-Wireless-Carplay-Touchscreen-Mirrorlink/dp/B0C23SNRTC
Probes and wires
LCD bare metal demo

https://github.com/orangecms/d1rgb/tree/cmp-hack
(forked from https://github.com/adamgreig/d1rgb)
Development setup
Development setup
Projects focusing on products

- OpenWrt, pfSense/OPNsense routers, network gear, WiFi
- Excellent OpenWrt wiki
- OpenIPC (network) cameras, lotsoftooling, tutorials, etc.
- OpenBMC, u-bmc board management controllers, remote OOB management
- Start a new one – pick u-root and cpu
- https://github.com/u-root/cpu
- https://github.com/orangecms/arm-cpu
- https://github.com/u-root/sidecore
Projects focusing on products

OpenWrt, pfSense/OPNsense
- routers, network gear, WiFi
- excellent OpenWrt wiki

OpenIPC (network) cameras

OpenBMC, u-bmc
- board management controllers
- remote OOB management

Start a new one - pick u-root and cpu

https://github.com/u-root/cpu
https://github.com/orangecms/arm-cpu
https://github.com/u-root/sidecore
Projects focusing on products

OpenWrt, pfSense/OPNsense
- routers, network gear, WiFi
- excellent OpenWrt wiki

OpenIPC
- (network) cameras
- lots of tooling, tutorials, etc

https://github.com/u-root/cpu
https://github.com/orangecms/arm-cpu
https://github.com/u-root/sidecore
Projects focusing on products

OpenWrt, pfSense/OPNsense
- routers, network gear, WiFi
- excellent OpenWrt wiki

OpenIPC
- (network) cameras
- lots of tooling, tutorials, etc

OpenBMC, u-bmc
- board management controllers
- remote OOB management
Projects focusing on products

- **OpenWrt, pfSense/OPNsense**
  - routers, network gear, WiFi
  - excellent OpenWrt wiki

- **OpenIPC**
  - (network) cameras
  - lots of tooling, tutorials, etc

- **OpenBMC, u-bmc**
  - board management controllers
  - remote OOB management

Start a new one - pick u-root and cpu

- [https://github.com/u-root/cpu](https://github.com/u-root/cpu)
- [https://github.com/orangecms/arm-cpu](https://github.com/orangecms/arm-cpu)
- [https://github.com/u-root/sidecore](https://github.com/u-root/sidecore)
Bringing up your device
Firmware vs OS

**U-Boot**
- configs in `configs/` - they determine the ARCH themselves
- device trees in `arch/$ARCH/dts/`
- boards in `board/$VENDOR/` - emphasis on SoC, but not consistently

**Linux**
- configs in `arch/$ARCH/configs/` - `$ARCH` must be provided by user
- device trees in `arch/$ARCH/boot/dts/` [$VENDOR/]
- board is described by firmware *and* own DTB, merged at runtime
Hardware Description: Device Tree

Standardization in progress; current version: 0.4

ADT must have a memory node—provided by firmware, usually.


Armtimer frequency must also be in DT, as I learned.

I simply put them in the kernel's DT, so I can do firmware without DT augmentation.

https://lore.kernel.org/linux-arm-kernel/25965de3-cc82-7fe6-6b3d-5754c329ac07@suse.de/
Hardware Description: Device Tree

[DeviceTree logo]

_devicetree_.org

Standardization in progress; current version: 0.4


Armtimer frequency must also be in DT, as I learned.

I simply put them in the kernel’s DT, so I can do firmware without DT augmentation.

https://lore.kernel.org/linux-arm-kernel/25965de3-cc82-7fe6-6b3d-5754c329ac07@suse.de/
Hardware Description: Device Tree

devicetree.org

Standardization in progress; current version: 0.4

A DT must have a memory node - provided by firmware, usually.

Hardware Description: Device Tree

**devicetree**
.org

Standardization in progress; current version: 0.4

A DT must have a memory node - provided by firmware, usually.


Arm timer frequency must also be in DT, as I learned.

I simply put them in the kernel’s DT, so I can do firmware without DT augmentation.

https://lore.kernel.org/linux-arm-kernel/25965de3-cc82-7fe6-6b3d-5754c329ac07@suse.de/
Kernel hacking

Early output

find indicator stoseehowfaryouget inearlyasm, directMMIOon serialforsinglecharoutput

▶ becarefulwithregisters-theyhavespecialmeaninginearlyasm

▶ doing a

will mess up the return address!

▶ debug.S reallyhandy, canprint2, 4, 8-digithexvaluesandASCII

Logs, logs, logs!

earlycon, figureitout

https://falstaff.agner.ch/2015/10/17/linux-earlyprintkearlycon-support-on-arm/

▶ for8250/16550: earlycon=uart, mmio32, $UARTBASE_ADDR

loglevel=8, initcall_debug, kernelconfigoptions
Early output

- find **indicators** to see how far you get
- in early asm, direct MMIO on serial for single char output
- `arch/$ARCH/kernel/head.S`
  - be careful with registers - they have special meaning in early asm
  - doing a `bl` will mess up the return address!
  - `debug.S` **really handy**, can print 2,4,8-digit hex values and ASCII

Logs, logs, logs!

For 8250/16550:
- `earlycon=uart,mmio32,$UARTBASE_ADDR`
- `loglevel=8`, `initcall_debug`, `kernelconfigoptions`
Kernel hacking

Early output
- find **indicators** to see how far you get
- in early asm, direct MMIO on serial for single char output
  - `arch/$ARCH/kernel/head.S`
    - ▶ be careful with registers - they have special meaning in early asm
    - ▶ doing a `bl` will mess up the return address!
    - ▶ `debug.S` really handy, can print 2,4,8-digit hex values and ASCII

Logs, logs, logs!
- `earlycon`, figure it out [https://falstaff.agner.ch/2015/10/17/linux-earlyprintkearlycon-support-on-arm/](https://falstaff.agner.ch/2015/10/17/linux-earlyprintkearlycon-support-on-arm/)
  - ▶ for 8250/16550: `earlycon=uart,mmio32,$UARTBASE_ADDR`
  - `loglevel=8, initcall_debug, kernel config options`
A little userland

build-arm32.sh

#!/bin/sh

set -e

export GOARCH=arm
CPIO="/tmp/u-root-$GOARCH.cpio"

# build a root fs using the embedded template
go run . -uroot-source . -o "$CPIO" embedded

# https://github.com/u-root/u-root/#compression
xz --check=crc32 -9 --lzma2=dict=1MiB --stdout "$CPIO" | \\n  dd conv=sync bs=512 of="$CPIO.xz"
Getting stuck

/platform: wait for supplier

In this case, I missed describing the power supply. It was a wrong guess anyway. More later.
Getting stuck

```bash
/# cat /sys/kernel/debug/devices_deferred
1c50000.ethernet  platform: wait for supplier
/soc/i2c@1c2ac00/pmic@34/regulators/dc1sw
```
Getting stuck

```
#!/ sys/kernel/debug/devices_deferred
1c50000.ethernet    platform: wait for supplier
/soc/i2c@1c2ac00/pmic@34/regulators/dc1sw
```

In this case, I missed describing the power supply.
Getting stuck

```
#/ cat /sys/kernel/debug/devices_deferred
1c50000.ethernet  platform: wait for supplier
/soc/i2c@1c2ac00/pmic@34/regulators/dc1sw
```

In this case, I missed describing the power supply.

It was a wrong guess anyway. More later.
Device Tree is nice, but…
Device Tree is nice, but…

The DT *could be checked at build time!*
Device Tree is nice, but…

The DT *could be checked at build time!*

Unless… the firmware is expected to provide (part of) it.

How about fallbacks?
Device Tree is nice, but...

The DT *could be checked at build time*!

Unless... the firmware is expected to provide (part of) it.

How about fallbacks?

**Solving Devicetree Issues, part 3.0**

Frank Rowand at ELCE 2016

https://www.youtube.com/watch?v=BDS6Hydtsx8

Device Tree is nice, but…

The DT *could be checked at build time!*

Unless… the firmware is expected to provide (part of) it.

How about fallbacks?

**Solving Devicetree Issues, part 3.0**

Frank Rowand at ELCE 2016

https://www.youtube.com/watch?v=BDS6Hydtsx8


Some great ideas which never landed upstream. Anyone?
Living the lie

DeviceTree is a tree—but your hardware is not!

Clocks, interrupts, GPIO pins, power supplies are all across.

Some entries in DT are just loose strings or references, e.g., phy-supply.

https://elinux.org/Device_Tree_Mysteries#Phandle

I've started building a device tree visualizer! :‑)
Living the lie

Device Tree is a tree - but your hardware is **not**!
Device Tree is a tree - but your hardware is **not**!

Clocks, interrupts, GPIO pins, power supplies are all across.
Living the lie

Device Tree is a tree - but your hardware is **not**!

Clocks, interrupts, GPIO pins, power supplies are all across.

Some entries in DT are just loose strings or references, e.g., `phy-supply`. 
Device Tree is a tree - but your hardware is not!

Clocks, interrupts, GPIO pins, power supplies are all across.

Some entries in DT are just loose strings or references, e.g., phy-supply.

https://elinux.org/Device_Tree_Mysteries#Phandle
Device Tree is a tree - but your hardware is not!

Clocks, interrupts, GPIO pins, power supplies are all across.

Some entries in DT are just loose strings or references, e.g., phy-supply.

https://elinux.org/Device_Tree_Mysteries#Phandle

I’ve started building a device tree visualizer! :-}
Eventually, it could turn into an interactive editor.
Eventually, it *could* turn into an interactive editor.
Tracing Components
Tracing Components

SoC platforms may use PMICs to supply power to components.
Tracing Components
Tracing Components

SoC platforms may use PMICs to supply power to components.
Thank you! :)
Follow Me

Daniel Maslowski

https://github.com/orangecms
https://twitter.com/orangecms
https://mastodon.social/@cyrevolt
https://youtube.com/@cyrevolt
https://twitch.tv/cyrevolt


License: CC BY 4.0 https://creativecommons.org/licenses/by/4.0/