GTA04 Boot with Device Tree

A migration Concept
Requirements

• we need 3 different image types

• Production Image (PI) (2 partitions with MLO to flash Xloader and U-Boot)

• Single/Standard Image (SI) image kernel + some rootfs (standard case)

• pure rootfs Image (RI) for NAND partition
Production (PI), Standard (SI) and ROOTFS Images (RI)

- **Production (PI)**: production, unbricking, upgrading
  - MLO / X-Loader
  - U-Boot
  - Boot-Splash
  - boot.scr

- **Standard (SI)**: (Single Partition)
  - NAND: X-Loader
  - U-Boot+Boot-Splash
  - boot.scr
  - kernel
  - ubifs

- **ROOTFS Images (RI)**: (Multipartition)
  - kernel + bootargs.scr
  - Debian
  - QtMoko
  - Replicant
  - QuantumSTEP
  - ...

---

(C) 2014, Golden Delicious Computers GmbH&Co. KG -- CC-BY-SA
Requirements

- upgrade must be possible without RS232 cable and U-Boot console commands
- downgrade to older boot loader should be possible
Requirements

• same Standard Image SD card must boot on L2804, L3704, L7004, ... (swap your system image and important files between devices)
The SI must be „universal“
Needs dynamic configuration to device

<table>
<thead>
<tr>
<th></th>
<th>MLO</th>
<th>U-Boot</th>
<th>Kernel</th>
<th>Rootfs</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2804</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3704</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L7004</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neo900</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What must be configured dynamic?

- Display (U-Boot, Kernel, rootfs)
- Touch parameters (Kernel, rootfs)
- Bootslpash images (U-Boot)
- LEDs (U-Boot, Kernel)
- interfaces (variety of GPIO assignment, e.g. for TRF7970, display)
U-Boot

- U-Boot is flashed to NAND (and only stored on PI)
- we have multiple versions - one for each device variant
- you should never flash the wrong PI :)
- can setup different pinmux, gpios, display for boot menu
Kernel

• Kernel is either flashed to NAND or in /ulimage or /boot/ulimage

• should be the same for any device!

• currently our U-Boot passes a mux= parameter to the kernel and the board file can then modify the display etc.

• example: mux=GTA04A3, mux=GTA04b2
**rootfs**

- the preferred method to probe for device variations is that user space checks if the kernel provides some APIs
- every measure must be taken to avoid to ask for the device model or version
- the optimal portable rootfs has NO dependencies (not even on screen sizes/resolutions)
rootfs exceptions

• xorg.conf - hard codes physical screen dimensions (in mm)

• touch screen - rotation, precalibration (could be moved to TSC driver i.e. DT parameters)

• wwan on/off GPIO (we could better hide the real GPIO by adding some rfkill driver to the kernel)

• is currently done by checking /proc/cmdline for the mux= value
Device Tree


• replaces (potentially dynamic) board file by a (statically compiled) device tree

• can be loaded by boot loader

• or can be appended to the kernel zImage
Device Tree

what is the problem?

• DT is not dynamic - we can’t check for the mux= variable!

• we need a different DT file (e.g. omap3-gta04+b2.dtb) for each device, i.e. we can’t append to an universal kernel

• Therefore U-Boot must load the right one

  • either from NAND (where it could be appended since NAND is a RI with separate kernel)

  • or choose one (of several) from the SI SD card

• we must ensure that SD card provides all relevant device tree binaries for some kernel for an SI
solutions: booting

- U-Boot knows the device it is running on (because we have device specific variants in NAND or on PI)
- U-Boot already provides the mux= variable - but that is „old style“
- we can initialize another environment variable:
  e.g. devicetree=omap3-gta04+b2
- the boot script can try to
  - locate and load some uImage and ${devicetree}.dtb file
  - or if no kernel is found, use the device tree in NAND for the kernel in NAND
solutions: flashing

• can we simply append the relevant DT by the flash-nand script?
• Yes, it could be possible if we have the mkimage tool on the production SD
• and we can extract the zImage by a script: http://buffalo.nas-central.org/wiki/How_to_Extract_an_uImage
  • dd if="${UIMAGE}" of="${TMPFILE}" ibs=64 skip=1 (header)
  • dd if="${TMPFILE}" of="${zImage}" ibs=8 skip=1 (ARM Mach type header)
• So it is possible for a kernel booted from SD to flash itself into NAND and append the required DT
solutions: distributing kernels

- we must provide all potential .dtb files in the boot directory
- so that they are available to boot a SI on all devices that are supported
- (minor) problem: growth of the number of files if we support more and more devices in the future
Migration plan

- U-Boot: provide devicetree= variable
- boot.scr: load ${devicetree}.dtb from the same location where ulimage was found
- boot.scr: use device tree from NAND if NAND kernel is booted
- kernel: provide ${devicetree}.dts files for all device variants we have
  - we may need to use .dtsi file(s) to organize common parts
  - file names must match constants compiled into U-Boot variants
- kernel build script: copy all relevant .dtb files to .../unstable
- kernel build script: add all relevant .dtb files to the .deb packages
- flash-nand: find a way to append the currently used device tree file to the currently used ulimage when flashing to NAND
  - or (big change!): add a new device-tree MTD partition to U-Boot and Kernel
- makesd (Pl): install all relevant device trees in /boot
- rootfs: replace check for mux= in /proc/cmdline with /proc/device-tree/model
- what else?
Migration Phases

• Phase 1: compile kernel with DT
• Phase 2: make U-Boot and boot.scr locate and load the correct DT
• Phase 3: make DT kernel boot
• Phase 4: adapt scripts for NAND flashing and configs