From first power-on to OS booting: software solutions for hardware issues

October 2016

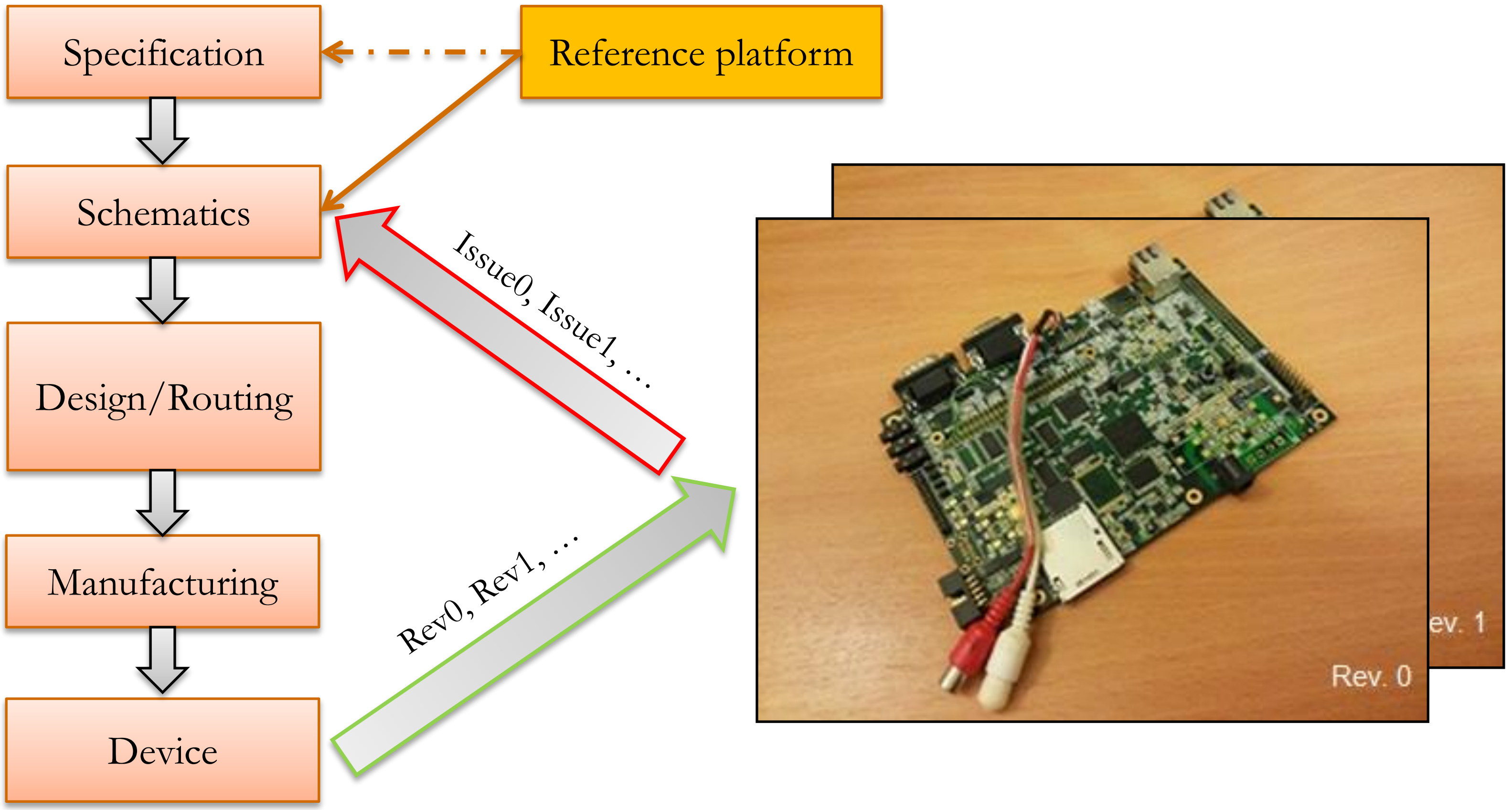
2016
CEE-SEC(R)
Software Engineering Conference in Russia

AURIGA®

Elite Software R&D Services
Since 1990

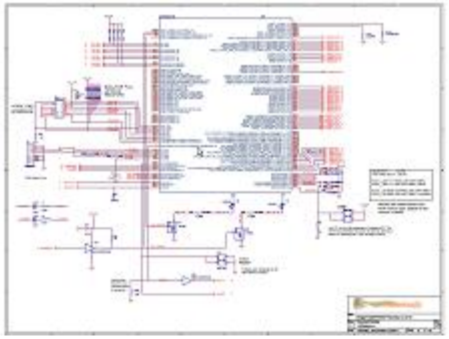
Device development

Typical design flow



In the beginning

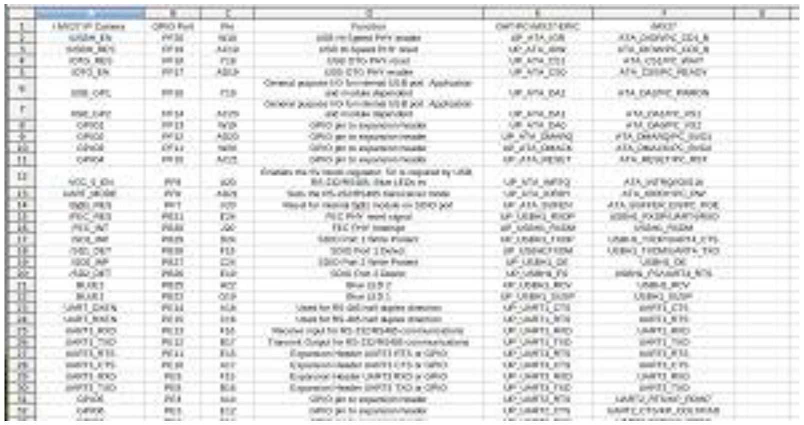
Reference platform



Device under development

("10 differences")

PINMUXes, GPIOs, CLKs, etc:



PowerOn

- Leds are on
- Voltages are OK
- Clocks are generated
- POR sequence is correct
- No smoke!

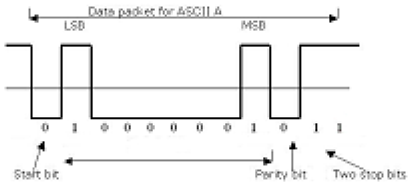
$\xrightarrow{\quad}$ Boot

Engineer’s tools

UART as “window into the world”

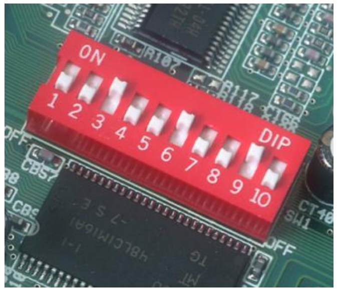
JTAG as “last hope”

GPIO bit-banging



System Boot

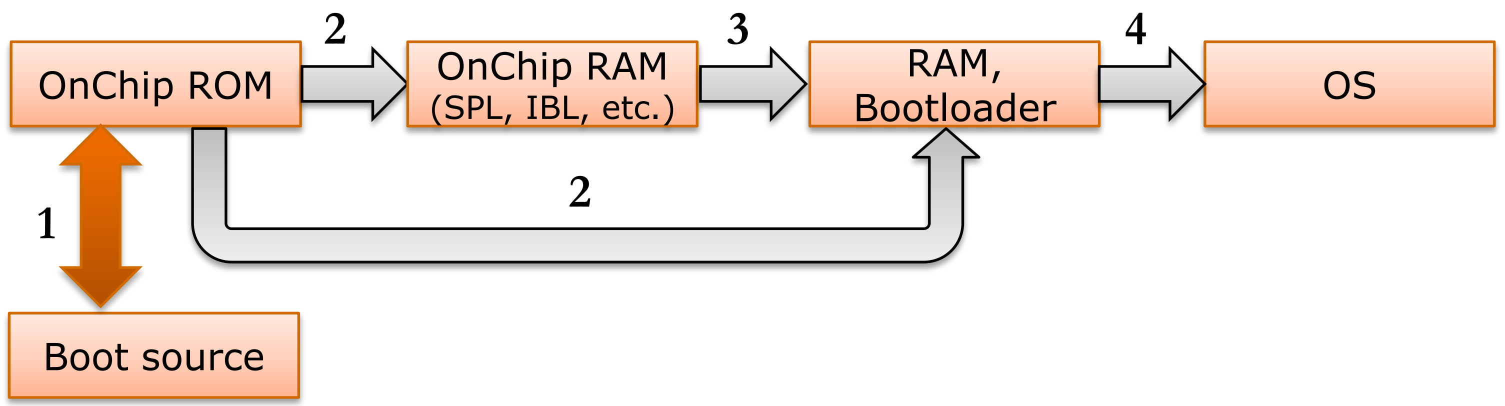
Boot mode



- NOR Flash
- NAND Flash
- SD/MMC/eMMC
- USB
- UART ("BOOTME"!)
- SPI/I2C (NOR, EEPROM)
- …

System Boot

Typical booting sequence



1. OnChip ROM detects boot source
2. OnChip ROM reads intermediate bootloader into OnChip RAM and transfers control to it
   Or
   OnChip ROM reads configuration into OnChip RAM, initializes RAM, loads and runs main bootloader
3. SPL initializes RAM, loads and runs main bootloader
4. Main bootloader loads OS

Intermediate bootloader

Common name

SPL – Secondary **P**rogram **L**oader

Other names (Vendor specific):

IBL (Intermediate Boot Loader)
UBL (User Boot Loader)
X-loader (External Loader)
…

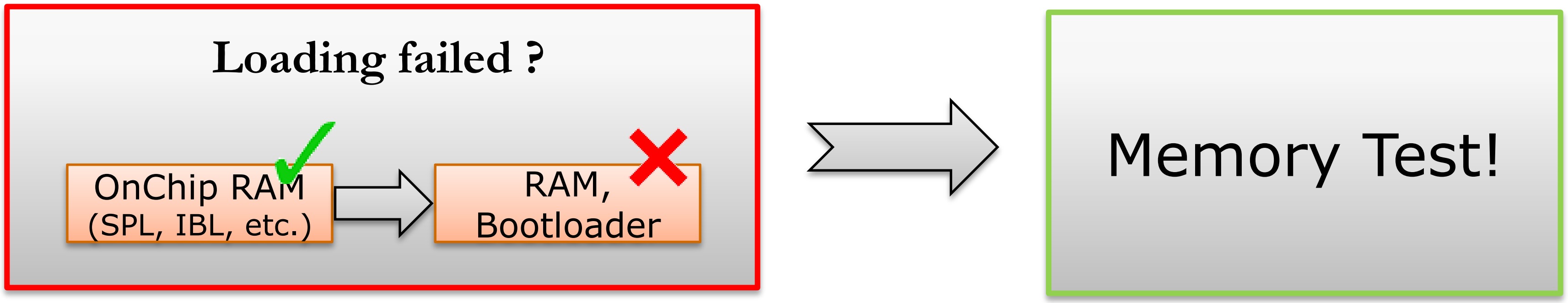
Features:

- Open source (Vendor provided)
- Executing in OnChip RAM
- Small size (OnChip RAM)

Main functions:

- Memory Controller and RAM initialization
- Loads main bootloader

RAM test



Classic Memory Test:

- Pattern (0x00000000, 0xFFFFFFFF, …)
- Walking '1'
- Walking '0'
- Address
- Word, halfword, byte

Implementation:

- SPL
- Standalone (OnChip RAM)
- Registers based (NOR flash)

Bootloader

Alternatives:

- U-Boot (ARM, x86, …)
- RedBoot (ARM, x86, …)
- PMON2000 (MIPS, PowerPC)
- PRIME (MIPS)
- …

Capabilities:

- Open source
- Simple configuring (GUI, menuconfig)
- Monitor functions (memory r/w, devices)
- Extensibility (new functionlality)
- Scripting (U-Boot, RedBoot)
- Client applications (HW tests)

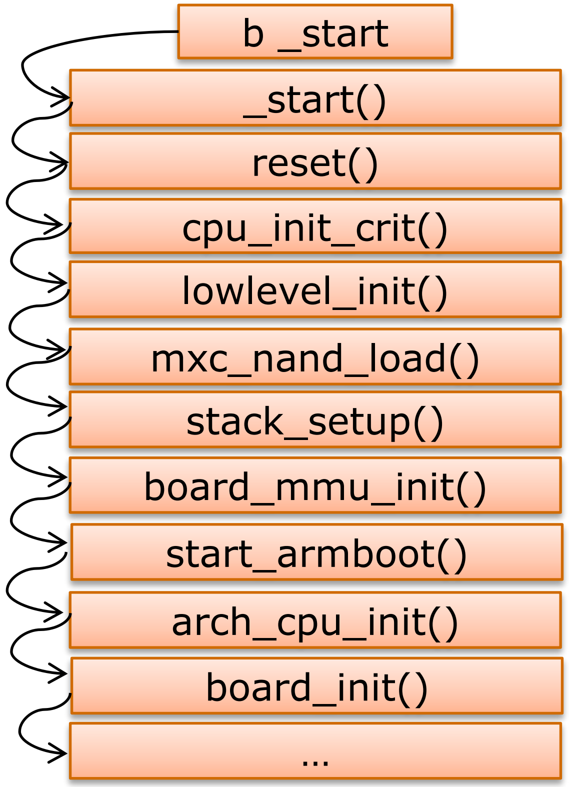
U-Boot: adaptation

Typical actions:

- Choose platform (board)
- Configure (make board_config)
- Build (make)
- Prepare bootable image (UART, MMC, …)
- First boot
- Check first boot results
- Adjust and reconfigure
- Debug
- Add new platform (board)

U-Boot: adaptation

U-Boot calls sequence partial breakdown:
(i.MX51 Babbage board)



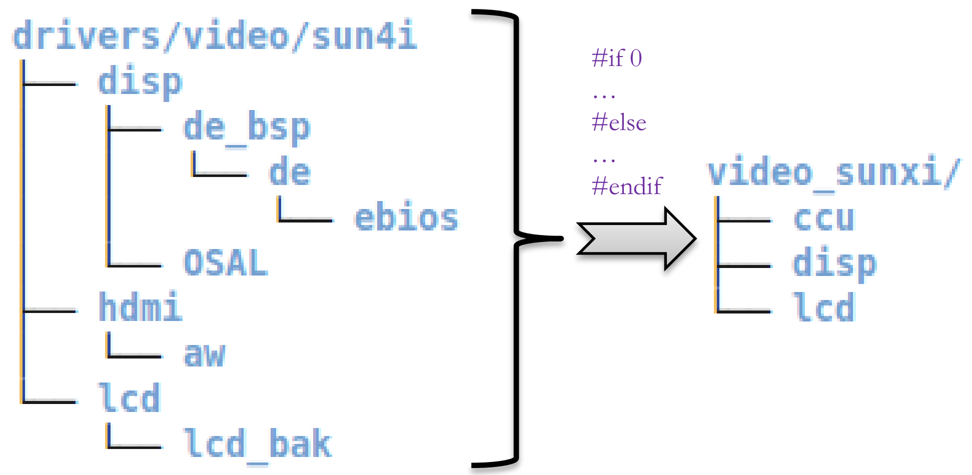
b _start	board/freescale/mx51_bbg/flash_header.S
_start()	cpu/arm_cortexa8/start.S
reset()	cpu/arm_cortexa8/start.S
cpu_init_crit()	cpu/arm_cortexa8/start.S
lowlevel_init()	board/freescale/mx51_bbg/lowlevel_init.S
mxc_nand_load()	cpu/arm_cortexa8/mx51/mxc_nand_load.S
stack_setup()	cpu/arm_cortexa8/start.S
board_mmu_init()	board/freescale/mx51_bbg.c
start_armboot()	lib_arm/board.c (init_fnc_t *init_sequence[] = {…})
arch_cpu_init()	cpu/arm_cortexa8/mx51/generic.c
board_init()	board/freescale/mx51_bbg/mx51_bbg.c
…	

U-Boot: enhancement

Add new driver

Allwinner A10 DISP driver
Linux kernel 3.0.8

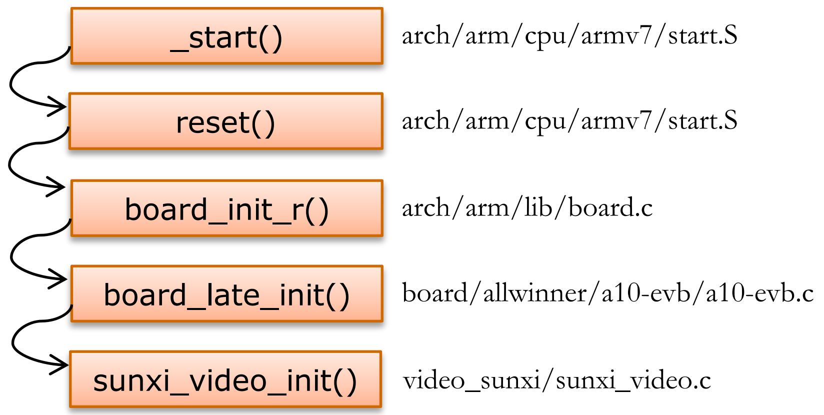
U-Boot



```
drivers/video/sun4i
├── disp
│   ├── de_bsp
│   │   └── de
│   │       └── ebios
│   └── OSAL
├── hdmi
│   └── aw
└── lcd
    └── lcd_bak
```

```
video_sunxi/
├── ccu
├── disp
└── lcd
```

Calls sequence



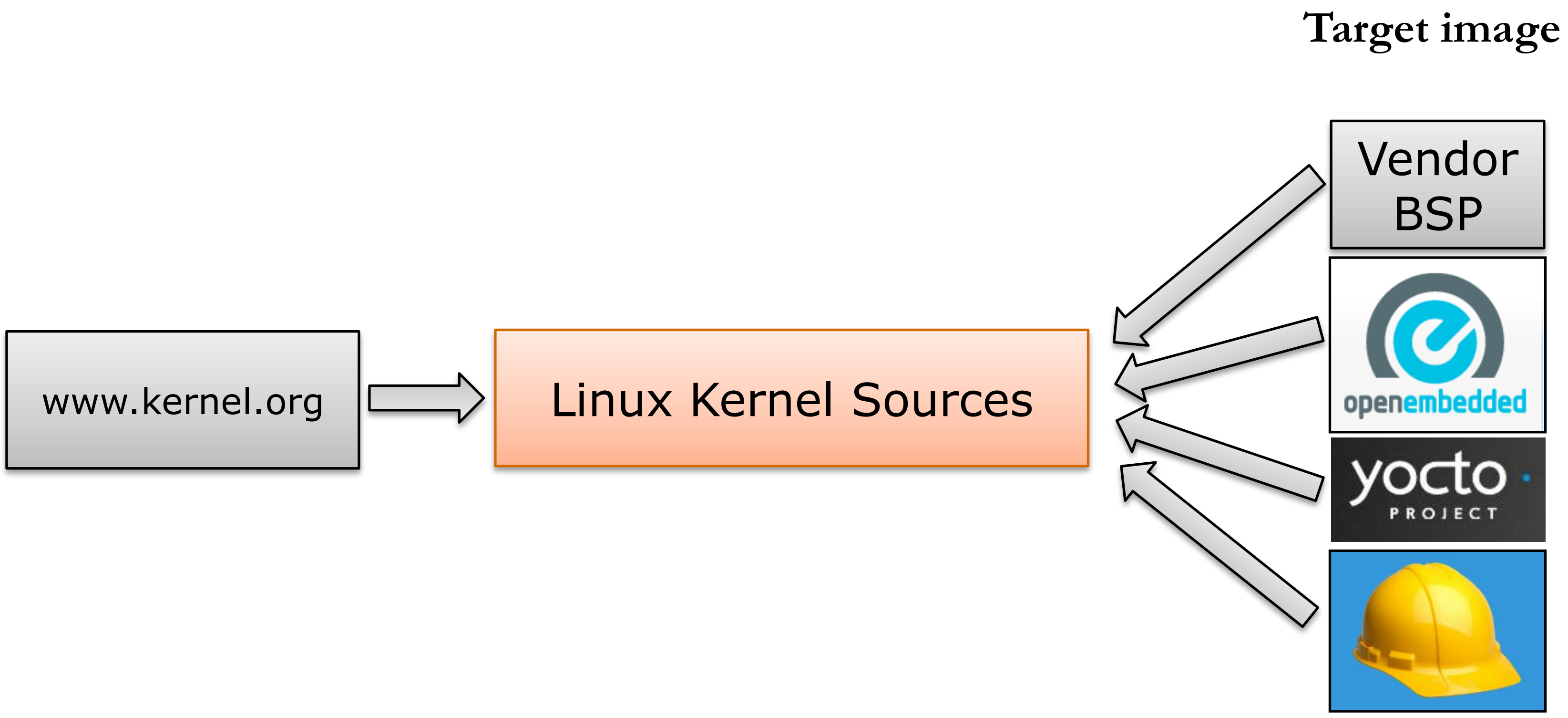
_start()	arch/arm/cpu/armv7/start.S
reset()	arch/arm/cpu/armv7/start.S
board_init_r()	arch/arm/lib/board.c
board_late_init()	board/allwinner/a10-evb/a10-evb.c
sunxi_video_init()	video_sunxi/sunxi_video.c

1MB (97 files) sources in total

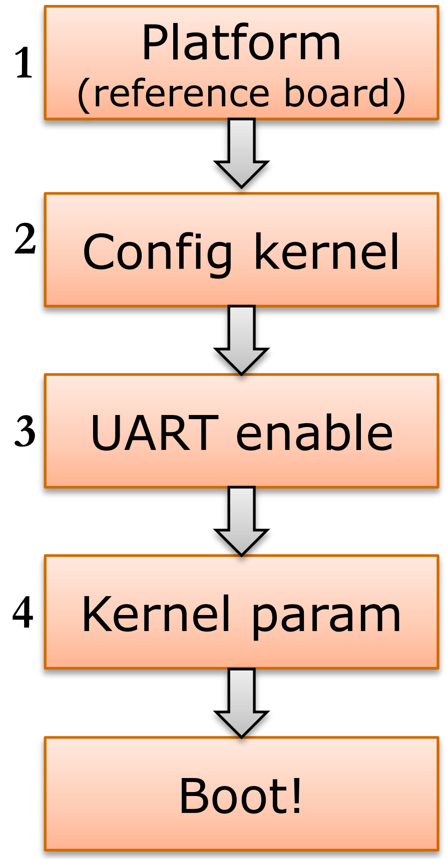
900KB (55 files) sources in total

OS: porting Linux

Kernel sources



OS: porting Linux



First steps

1. arch/arm/mach-<soc>,
   <soc> = imx, davinci, pxa, …
2. make menuconfig,
   Kernel hacking ->
   Kernel low-level debugging port/functions
   Early printk
3. Update platform source (2.6)
   or
   Update DTS (3.0, …):
   arch/arm/boot/dts/<soc>-<platform>.dts
4. earlyprintk=tty<uart_name><num>,
   <uart_name> = S, O, mxc, …
   <num> = 0, 1, 2, …
   boardid: arch/arm/tools/mach-types (2.6)

OS: porting Linux

Kernel booting issues

No kernel messages

- Check UART settings and configuration platform (2.6) / DTS (3.0, …)

- Enable early debug:
  arch/arm/boot/compressed/**head.S**
  arch/arm/mach-<soc>/include/mach/
  debug-macro.S (2.6)
  or
  arch/arm/boot/compressed/**debug.S** (3.0, …)
  Макросы:
  addruart
  senduart
  busyuart
  waituart

OS: porting Linux

Kernel successfully booted

Kernel is booted!

- Load simple target image
- Create custom platform (finalizing changes)
- Build and load new target images (X-Window, Gnome, …)

Linux: kernel backporting

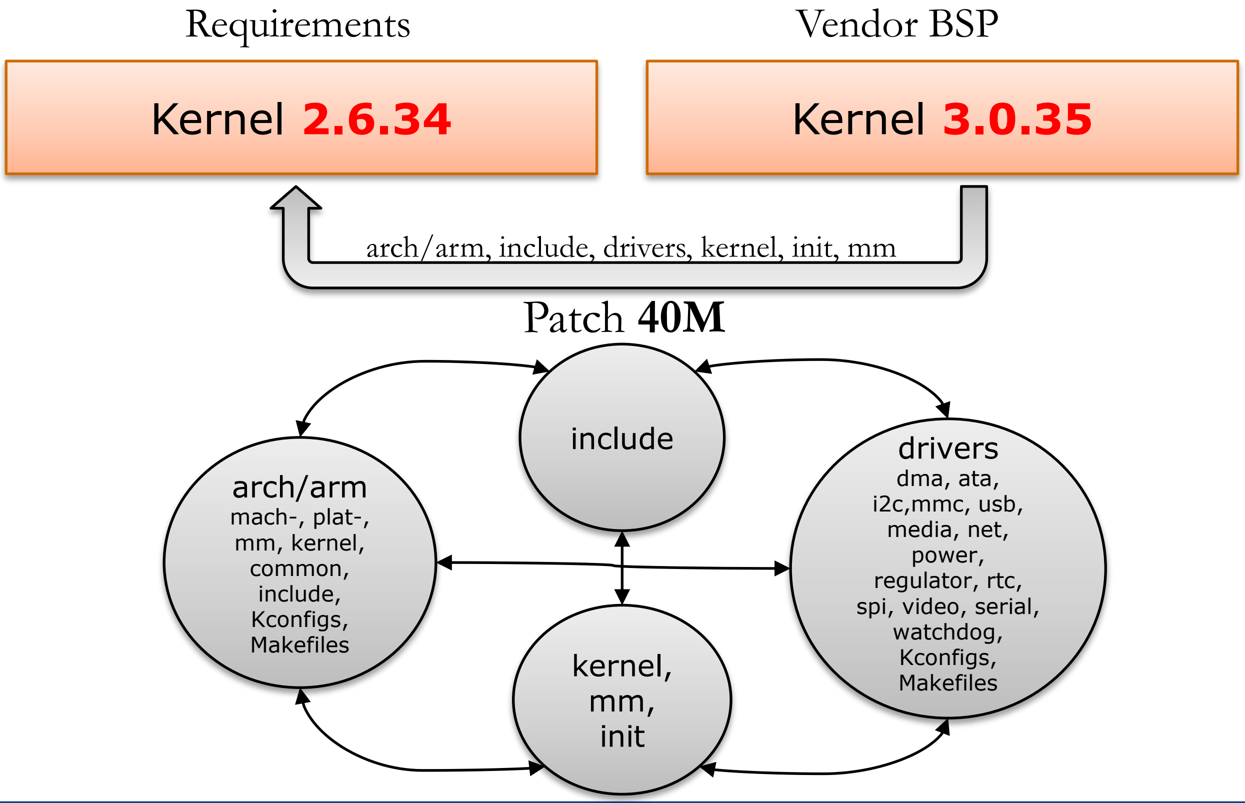
Reasons ?

- Customer requirements (particular version is needed)
- Support (driver updates)
- Security (fix vulnerabilities)
- ???

Linux: kernel backporting

From 3.0.35 to 2.6.34

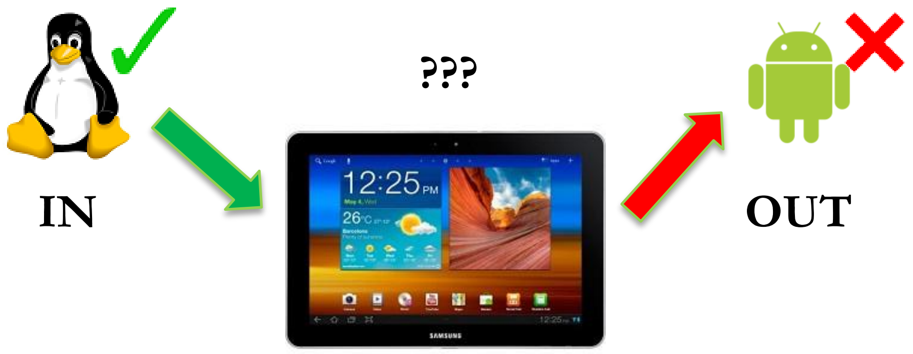
i.MX 6Dual based board



Requirements
Vendor BSP
Kernel 2.6.34
Kernel 3.0.35
arch/arm, include, drivers, kernel, init, mm
Patch 40M
include
arch/arm
mach-, plat-, mm, kernel, common, include, Kconfigs, Makefiles
drivers
dma, ata, i2c,mmc, usb, media, net, power, regulator, rtc, spi, video, serial, watchdog, Kconfigs, Makefiles
kernel, mm, init

OS: Linux instead of Android ?

The task



Would it be possible to install Linux instead of stock Android ?

Specific software

Requirements:

- Linux (any distro)
- GUI (X11)

OS: Linux instead of Android ?

Solution

30-pin connector

?

- Interface 0 ?
- Interface 1 ?
- …
- Interface n ?

!

- **USB OTG !**
- …
- **UART RXD/TXD !**
- …

USB host-USB client-UART cable

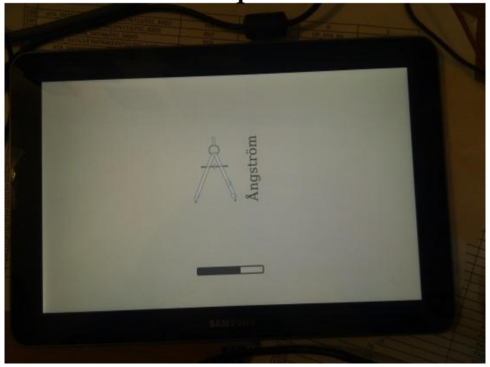
Steps

1. UART + null-modem cable
2. Boot to Recovery
3. Root !
4. mmcblk0p0, mmcblk0p1, …
5. USB flash drive
6. dd if=/dev/block/mmcblk0 of=/usbdrive/emmc.img
7. emmc.img => GPT => p0, p1, p2, … => p2 boot image (kernel + ramdisk) !
8. Download kernel sources (vendor provided)
9. OpenEmbedded + kernel sources => bzImage + rootfs
10. mkbootimg: bzImage + ramdisk => bootimg
11. UART, USB flash drive, Recovery: dd if=/usbdrive/bootimg of=/dev/block/mmcblk0p2
12. Kernel is booted !

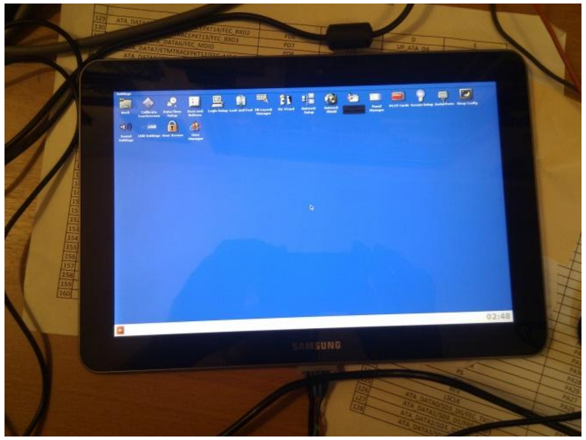
OS: Linux instead of Android ?

Results

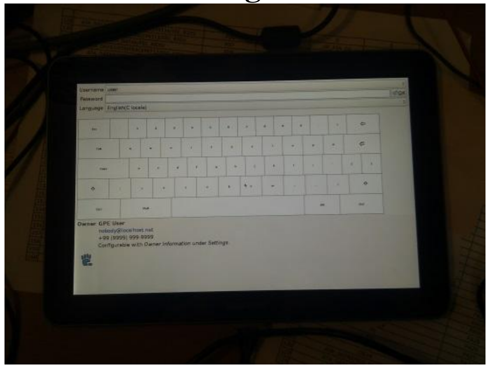
Boot splash



GPE



Login



Features

- Angstrom distro
- x11-gpe-image
- No multitouch is supported (modified TS-driver)
- Rootfs on USB drive (root=/dev/sda1)

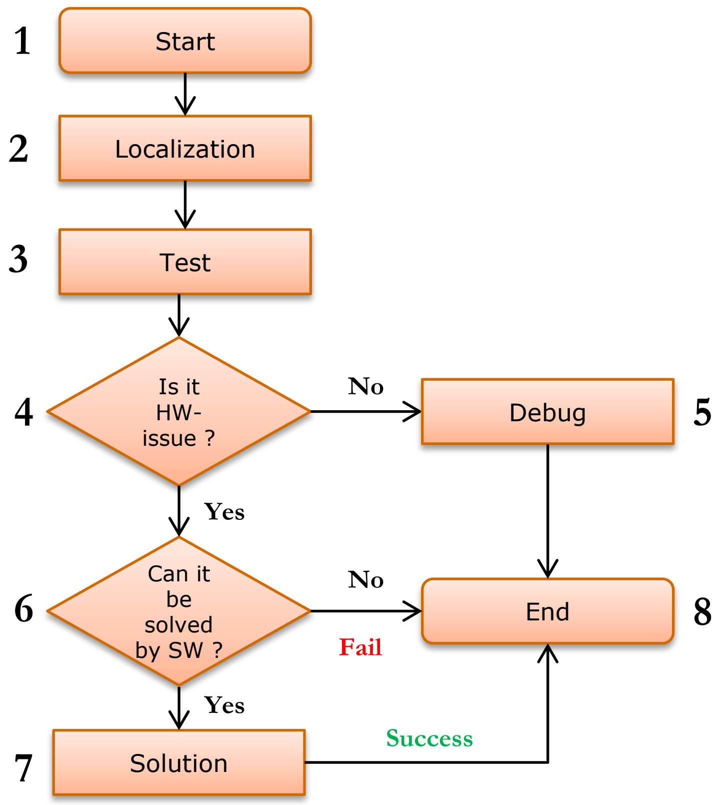
Hardware issues

Possible causes

- Design errors
- Routing errors
- Mounting issues
- Component defects
- Components errors (Errata, workarounds)
- ~~Physical damages~~
- ~~Incompatible conditions~~

Hardware issues

Common approach:
troubleshooting and looking for solution



1. Suspected HW-issue
2. Localization failing source code.
3. HW test
   ("three-lines program").
4. HW issue confirmed ?
5. No.
   Caused by SW.
6. Yes.
   Can be bypassed by SW ?
7. Yes.
   Working out solution and add it to sources
8. Further accordingly to results
   (Fail or Success).

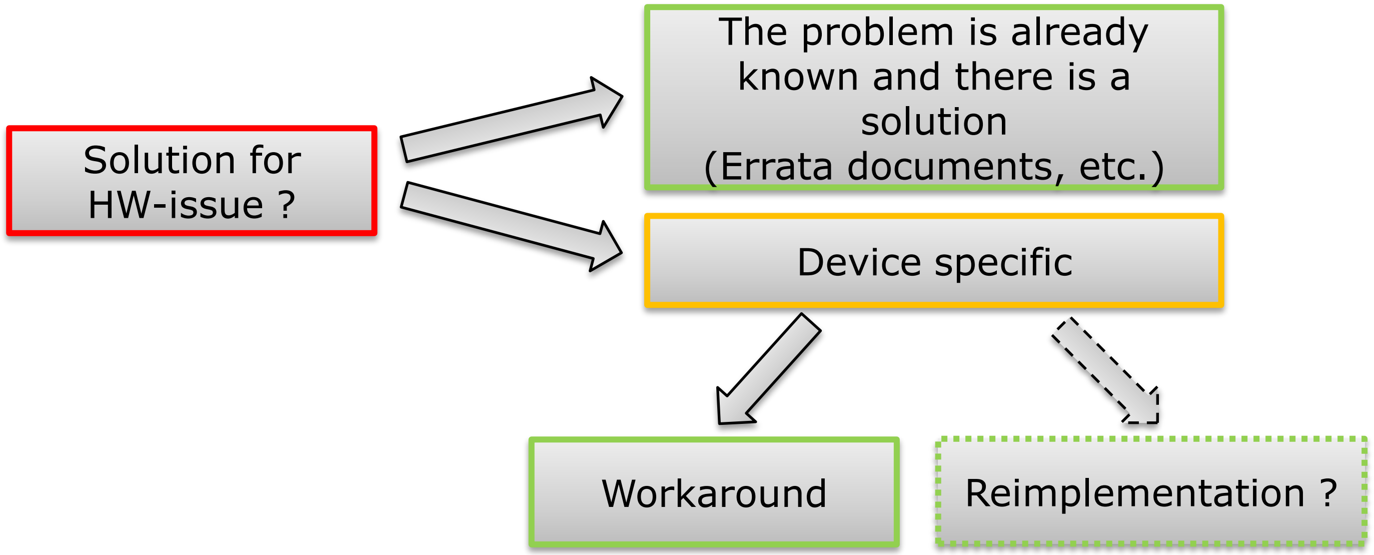
Hardware issues

Software solution

Workaround

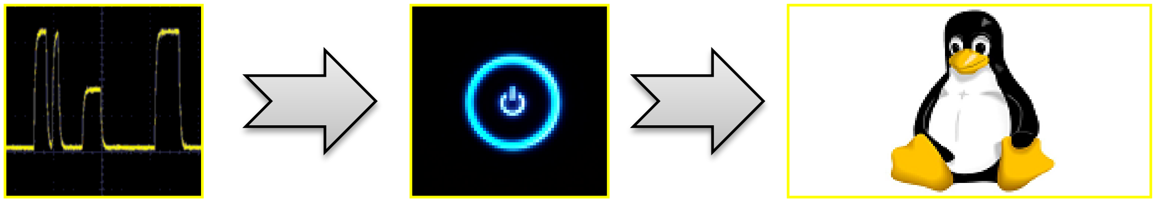
From Wikipedia, the free encyclopedia

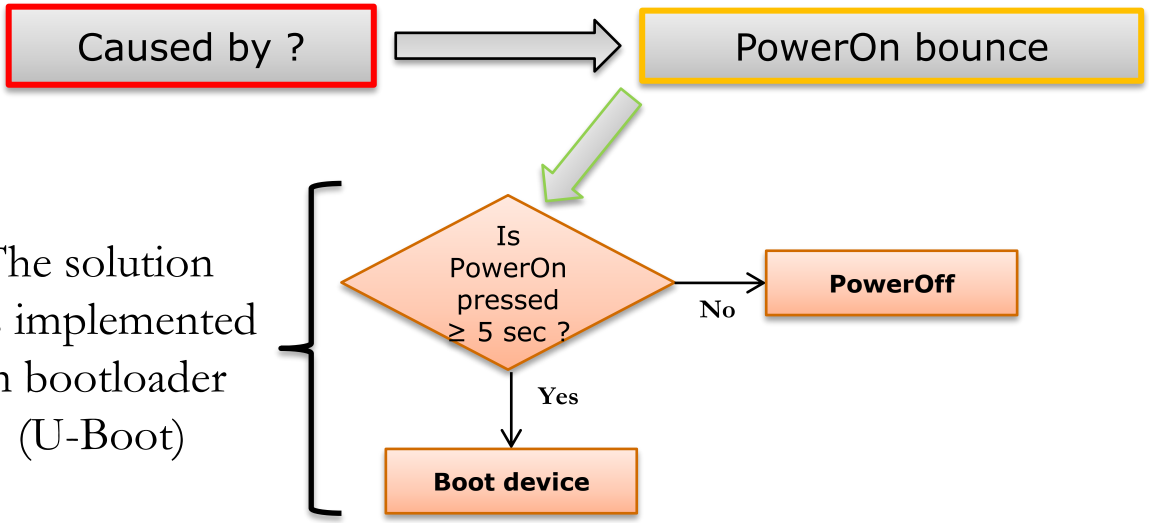
A workaround is a bypass of a recognized problem in a system.
A workaround is typically a temporary fix that implies that a genuine solution to the problem is needed.
But workarounds are frequently as creative as true solutions, involving outside the box thinking in their creation.


Solution for HW-issue ?
The problem is already known and there is a solution (Errata documents, etc.)
Device specific
Workaround
Reimplementation ?

Software solution: workaround

Spontaneous device power on

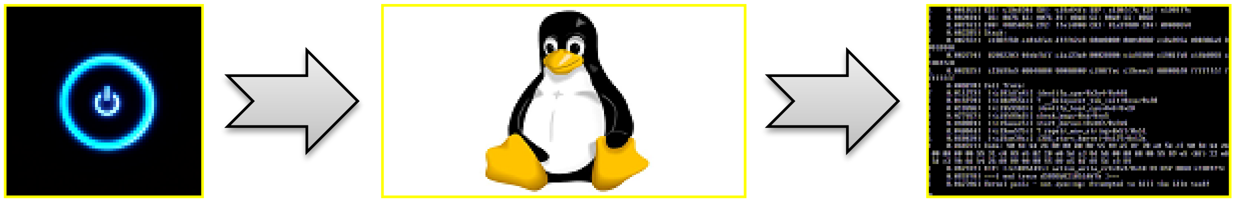


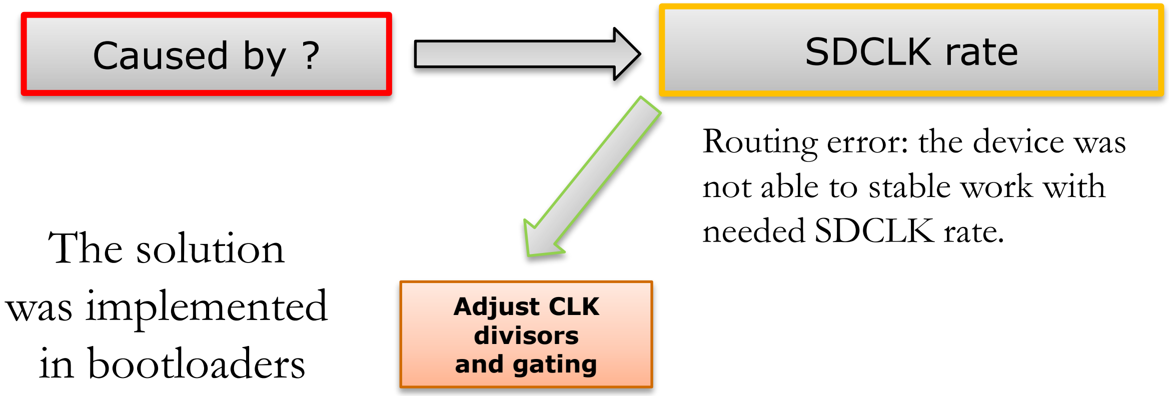


The solution was implemented in bootloader (U-Boot)

Software solution: workaround

Unstable device behavior



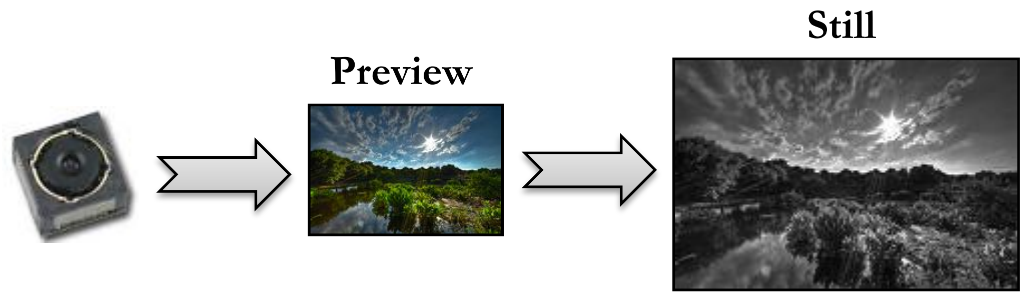


Routing error: the device was not able to stable work with needed SDCLK rate.

The solution was implemented in bootloaders (SPL, proprietary)

Software solution: workaround

CMOS camera1: black and white photos



Caused by ? → MasterClock (MCLK) rate

Design error: the device cannot provide camera with needed MCLK rate (it is impossible to find correct divisors values).

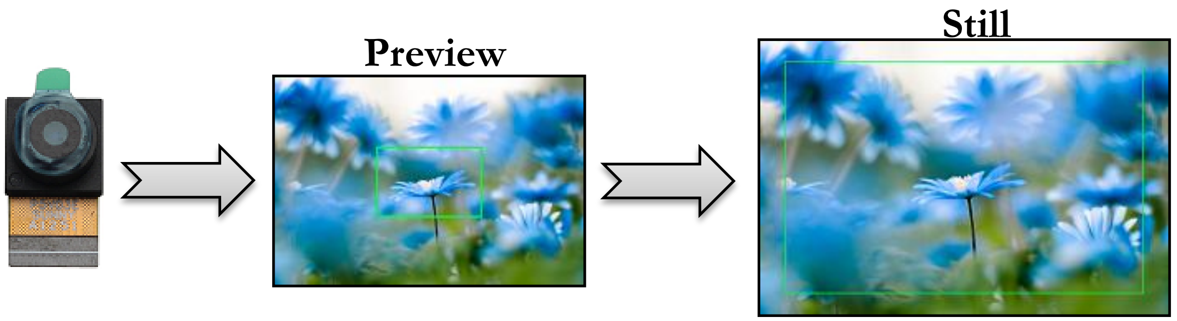
Adjust CLK divisors and gating (decrease clock rate)

The solution was implemented in driver.

Fixed by frequency rate decreasing that in turn caused FrameRate decreasing in Preview mode. (~10 fps).

Software solution: workaround

CMOS Camera2: autofocus frame is present on the photos



Caused by ? → Sensor firmware error

Autofocus frame is occasionally emerged on the photos despite of explicitly executed command “Hide frame” in accordance with Application Notes.

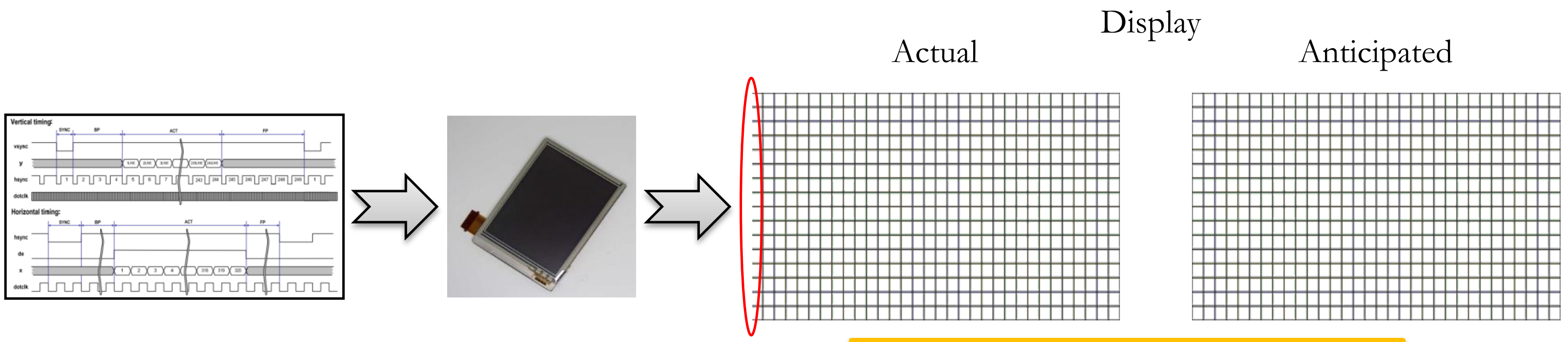
The solution was implemented in driver

Fixed by setting autofocus frame Hsize and Vsize to 0 via found undocummented registers

The solution (needed registers) was found through reverse-engineering of Sensor Devkit SWtools.

Software solution: workaround

LCD: the leftmost column issue



Caused by ? → HSyncStart, HSyncWidth, HTotal ?

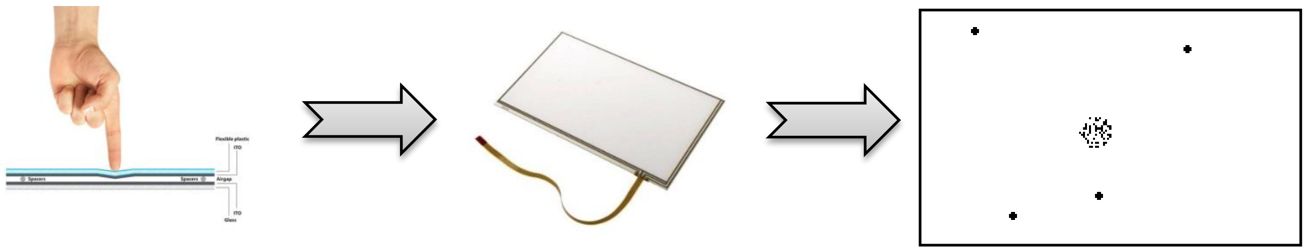
The needed values of horizontal synchronization parameters to correctly display leftmost column weren't eventually found. Presumable root cause is incompatibility of particular TFT LCD and videocontroller.

The solution implemented in driver

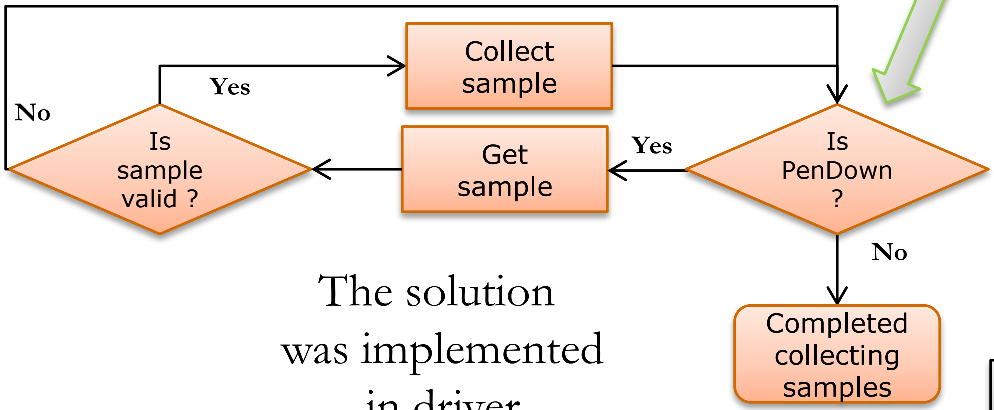
Fixed by filling framebuffer data in accordance with actual image size

Software solution: workaround

Resistive touchscreen: false pressings

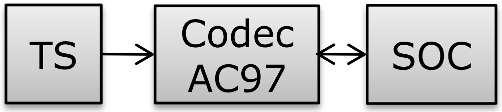


Caused by ? → Touchscreen feature ?



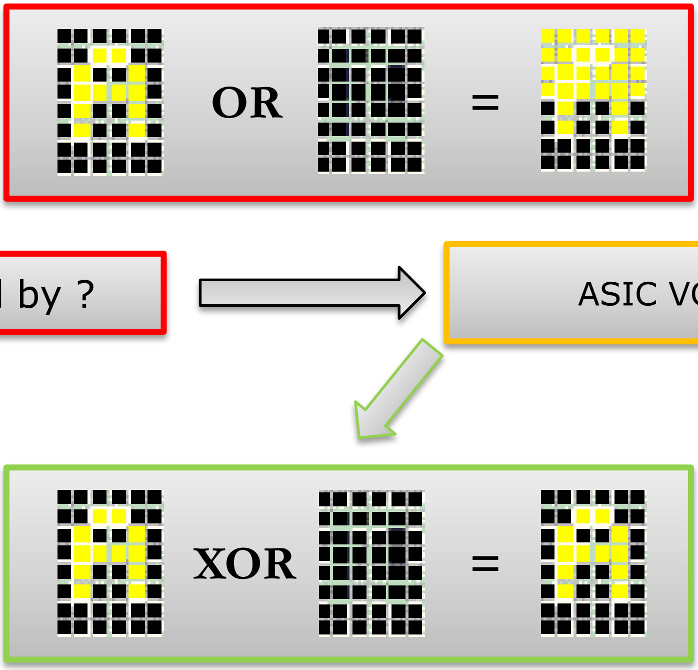
The solution was implemented in driver

False pressings were successfully filtered out via checking of distance between points.
Presumable root cause of such behavior is a feature of particular touchscreen connected as follows:



Software solution: reimplementation

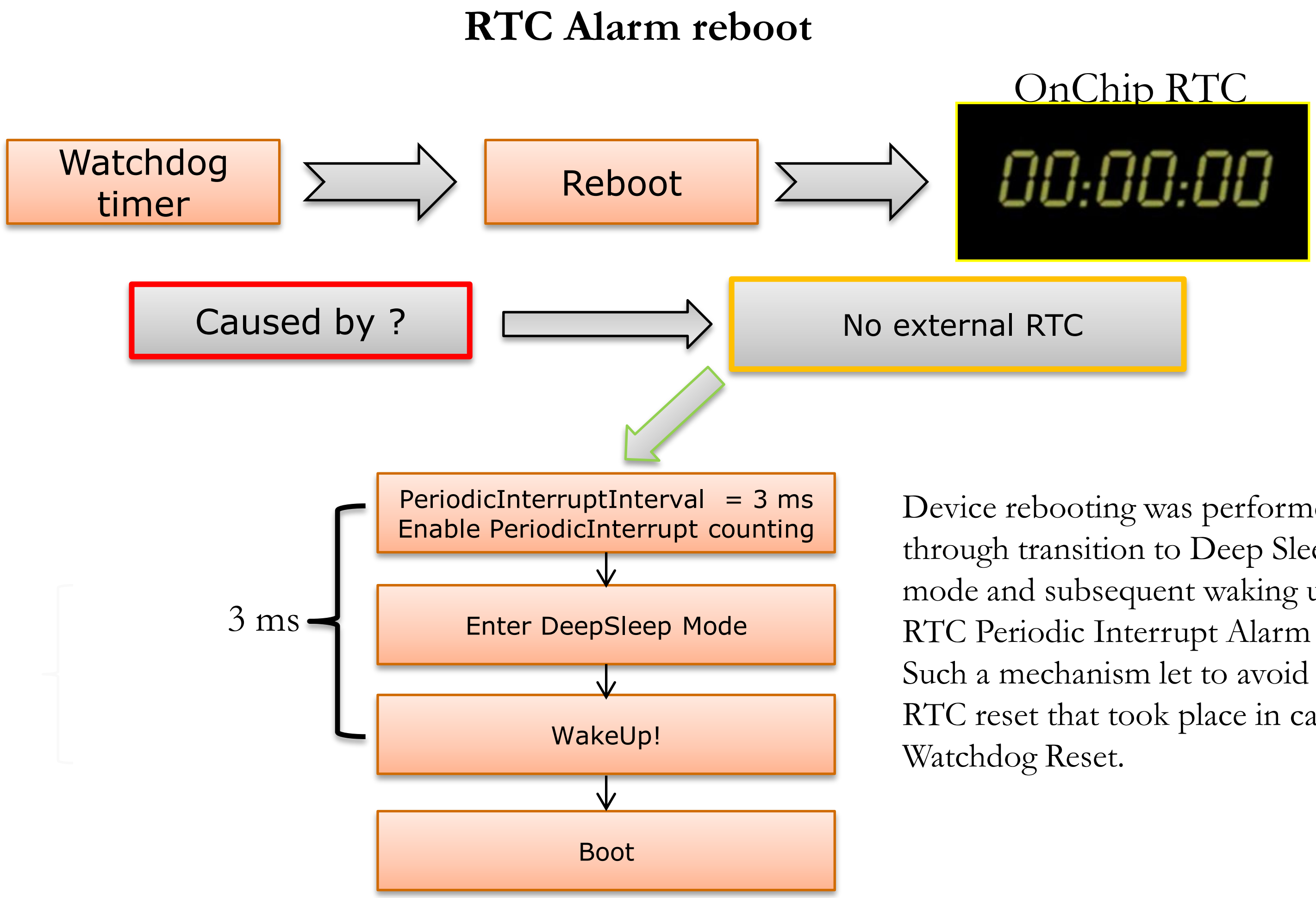
ASIC video controller:
incorrectly working hardware BitBlt with OR



The solution
was implemented
in driver

Symbols displaying was implemented with hardware BitBlt with XOR.

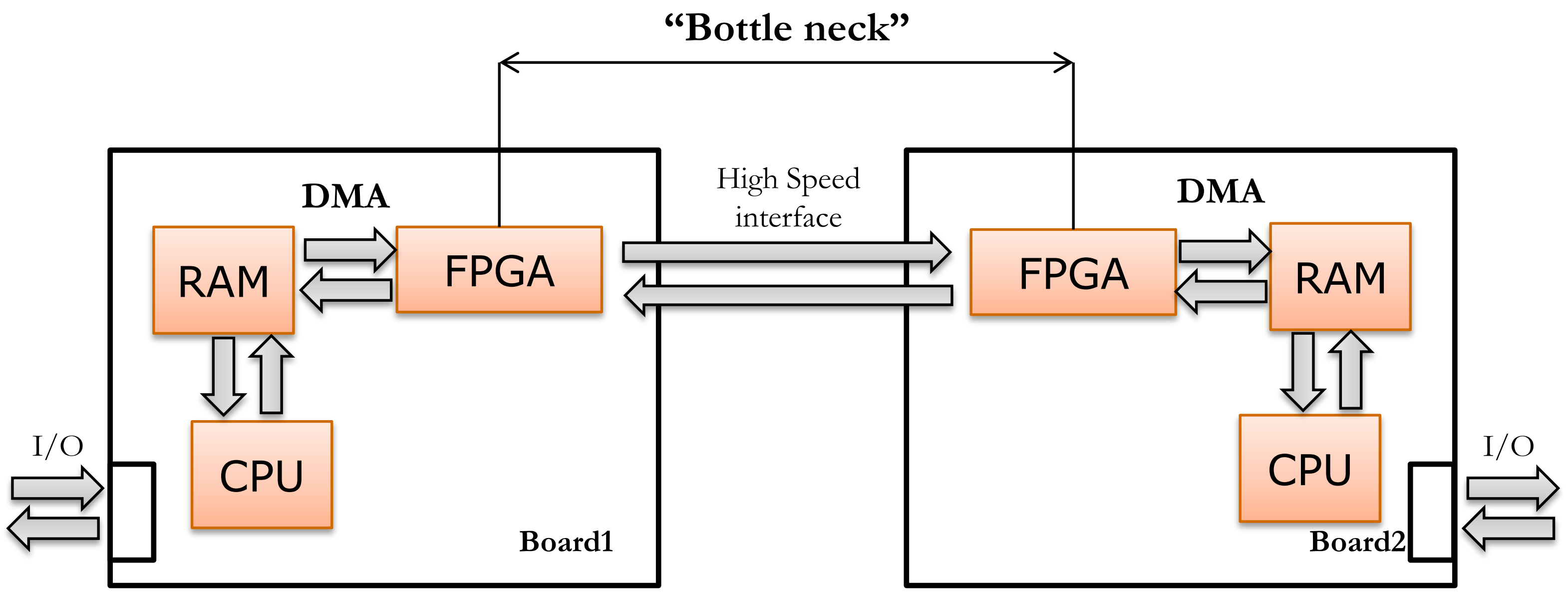
Software solution: reimplementation



Device rebooting was performed through transition to Deep Sleep mode and subsequent waking up on RTC Periodic Interrupt Alarm event. Such a mechanism let to avoid onchip RTC reset that took place in case of Watchdog Reset.

Software solution: not found

About design errors



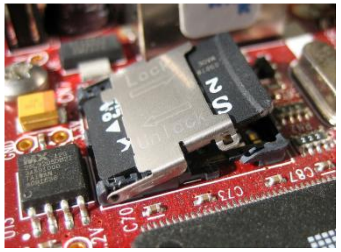
Due to erroneous design actual traffic rate turned out to be half the level of anticipated.
The bottle neck was FPGA <-> FPGA section.
This issue doesn't have a software solution.

Hardware issue ?

SD HC vs SD Host 1.01

SD Spec. 2.0

→


SD Host controller,
SD Spec. 1.01

→

?

CMD8,
Set Response Format
as R1/R6

→

It works!

This is not hardware issue!
In accordance with SD Spec. 2.0 SDHC cards shall execute CMD8 (Send Interface Condition) after CMD0 (Go Idle State) and before ACMD41 (Send Operation Condition) during card initialization.
Minor driver fix allowed to work with SDHC card on SD Spec. 1.01 host controller.

Contacts

AURIGA®

Thank You

Auriga, USA
92 Potter Rd, Ste. 1
Wilton, NH 03086, USA
Phone: +1 (866) 645-1119
Fax: +1 (603) 386-6097
info@auriga.com
www.auriga.com

Auriga, Russia
125 Varshavskoe Shosse, Unit 16A,
Moscow, 117587
Tel:+7 (495) 713-9900
Fax:+7 (495) 939-0300
info@auriga.com
www.auriga.com