Bootloader

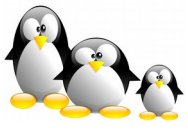
EIM/INFM

Frank Erdrich
frank.erdrich@emtrion.de

Ergänzung VL1 - Hardware

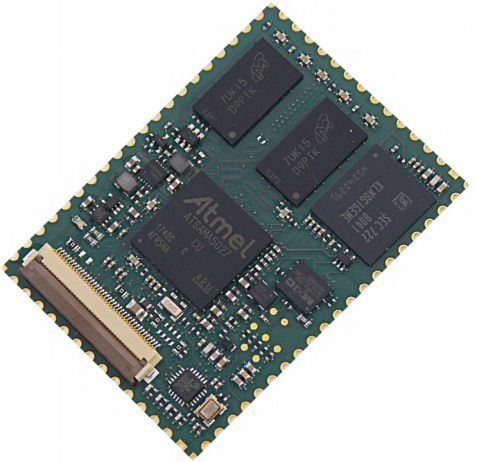
- Wie die Hardware auswählen?

- Anforderungen erfassen:
  - Rechenleistung
    - Anzahl CPU Kerne
    - CPU-Takt
    - 32bit vs. 64 bit



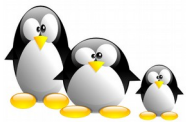
Ergänzung VL1 - Hardware

- Speicher
  - RAM
  - ROM (NAND, NOR, eMMC)
- Peripherie
  - Schnittstellen wie I²C, SPI, ADC, DAC, Ethernet, Funkstandards, ...
- Headless vs. Displayunit
- Grafikbeschleunigung?



Ergänzung VL1 - Hardware

- Spezielle Peripherie wie CNN Beschleuniger?
- Echtzeit?
  - Evtl. Zusätzlichen FPGA
  - Heterogenes System
- Security?
- Ausfallsicherheit?
- USV?



Ergänzung VL1 - Hardware

- Akkubetrieb? Stromverbrauch...
- Einsatzbereich
  - Temperaturbereich/Klima
  - Schockresistent?

- Bei Hardwareauswahl Matrix bilden zwischen Anforderungen und Hardwaresystemen.

Ergänzung VL1 - Hardware

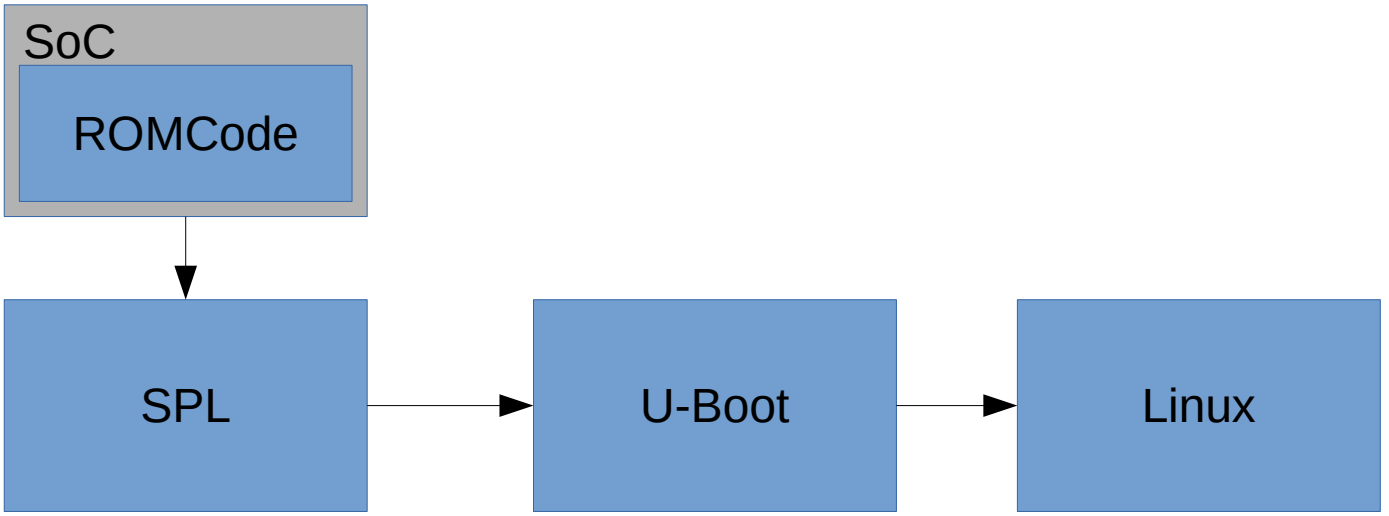
	HW A	HW B	HW C	...
Quad Core	✔ 4 Core	✘ 2 Core	✔ 4 Core	...
>= 1GB RAM	✘ 512 MB	✔ 1 GB	✔ 1GB	...
CNN IP	...	...	...	...
Bluetooth	...	...	...	...
...	...	...	...	...
0 – 70 °C	...	...	...	...
< 2W	...	...	...	...
Preis				

Das U-Boot

- Entwickelt von Wolfgang Denk (DENX)
- Open Source
- Communitygetrieben

Bootloader

- ROMCode (ROMBoot)
- SPL (Secondary Program Loader)
- U-Boot



http://www.denx.de/wiki/U-Boot/WebHome

Bootloader

Bootstage	Terminology #1	Terminology #2	Actual Programname
1	Primary Program Loader		ROM code
2	Secondary Program Loader	1st stage bootloader	U-Boot SPL
3		2nd stage bootloader	U-Boot
4			kernel

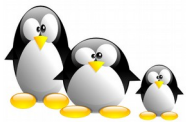
https://stackoverflow.com/questions/31244862/what-is-the-use-of-spl-secondary-program-loader

ROMCode

- Vom SoC-Hersteller implementiert
- Unveränderlich im SoC
- Teilweise über PULL-Up/-Down Widerstände steuerbar (Bootmedien)
- Sucht auf verschiedenen Bootmedien nach nächster Ausführungseinheit

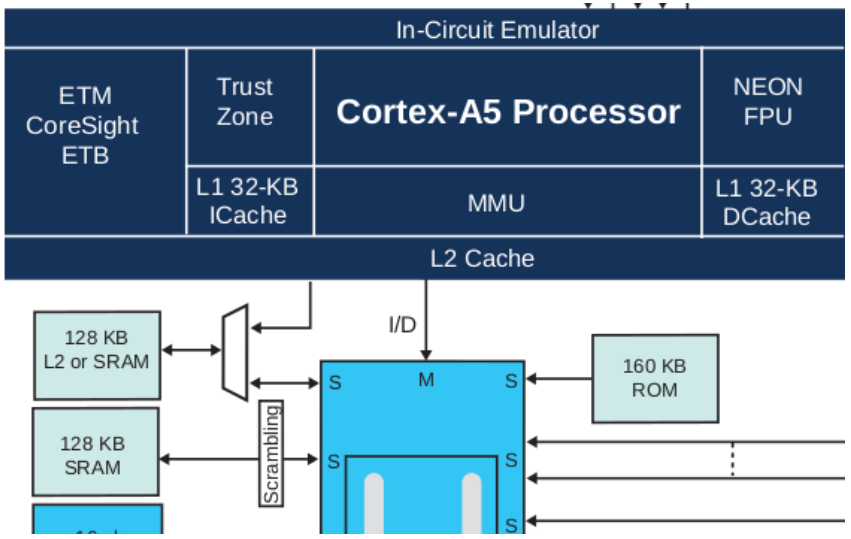
ROMCode

- Verschiedene Bootmedien
  - NOR-Flash
  - NAND-Flash
  - eMMC/SD-Card
- Unter Umständen mit Monitorprogramm (SAM-BA bei Microchip)
- SecureBoot

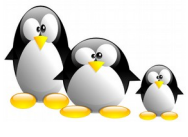


Warum SPL?

- Hardware uninitialisiert
- Kleiner interner SRAM steht zur Verfügung
- U-Boot zu groß für internen SRAM



https://www.uni-muenster.de/ZIV.MathiasGrote/linux/Dateisystem.html

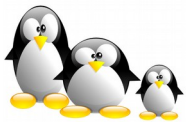


Aufgabe SPL

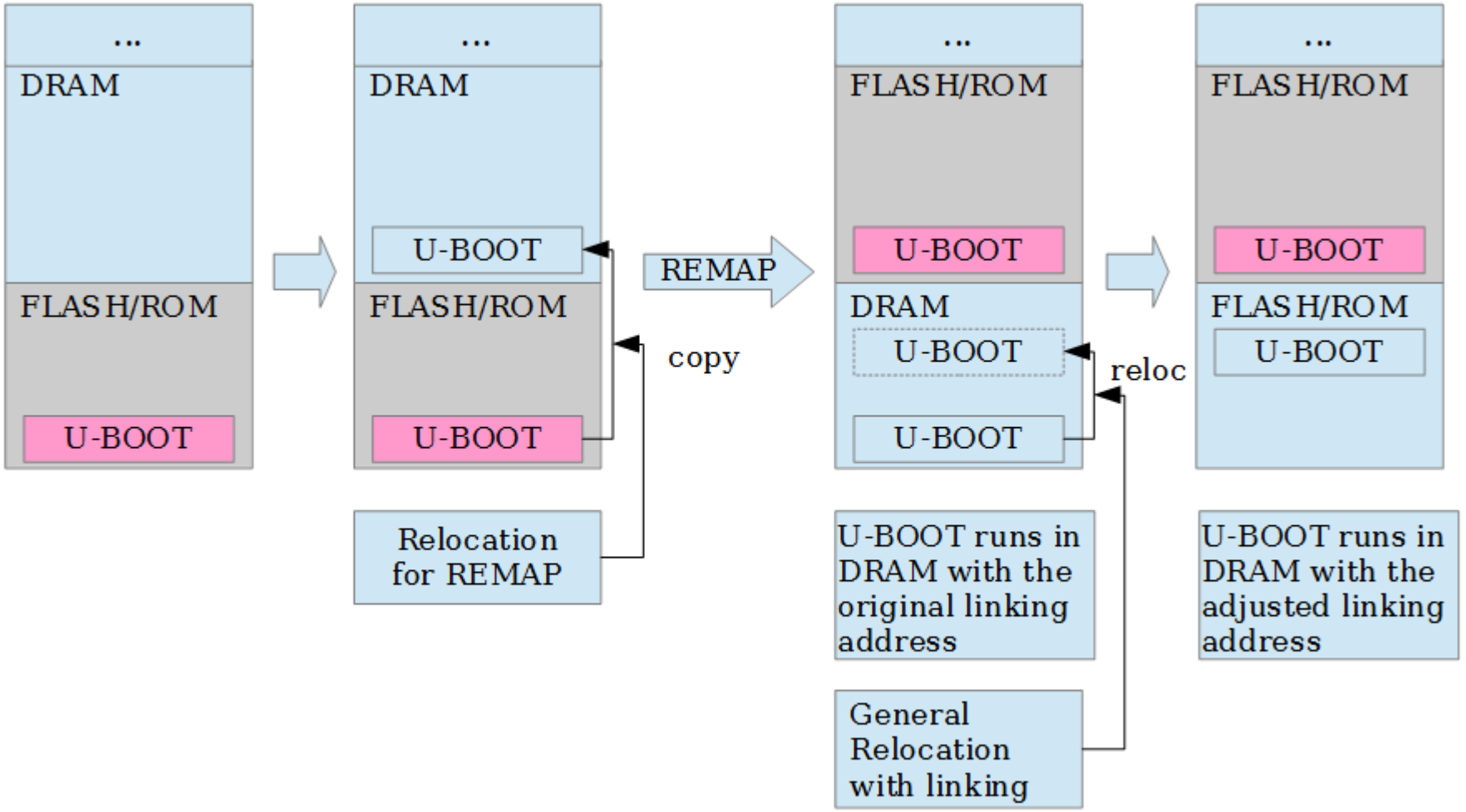
- Clocks initialisieren
  - CPU Clock
  - Peripherie-Clock für RAM
- DRAM initialisieren
- U-Boot laden und starten

U-Boot

- Übernimmt die Ausführung vom SPL
- Relokiert sich selbst
  - Kopiert sich an das Ende des DRAMs
  - Biegt alle internen Referenzen um (Funktionszeiger, Speicherbereiche, ...)
- Aktiviert weitere Peripherie
  - Netzwerk
  - Display
  - ...



U-Boot Relokation



U-Boot

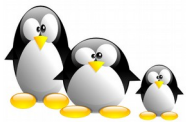
- Führt gegebenenfalls diverse Startscripte aus
- Lädt den Linux-Kernel, InitRAMFS und DeviceTree von definiertem Speicher
- Startet Linux und übergibt die CPU

U-Boot Features

- Unterstützung für viele SoCs
- Treiber für fast jede Peripherie
- Dateisystemunterstützung
  - Ext2/3/4
  - JFFS
  - UBIFS
  - ...

U-Boot Features

- Netzwerk
  - Booten über Netzwerk
  - DHCP, TFTP, NFS
- Serielle Debug-Schnittstelle
- Displayunterstützung mit Splashscreen
- Shell mit Scripting
- Environment

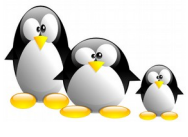


U-Boot Environment

- Speicher für User-Defined Settings
- Scriptspeicher
- Redundantes Environment möglich
  - Failsave bei Fehler
- Verschlüsselbar

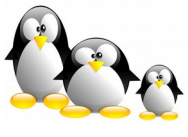
U-Boot Shell

- hush-shell
- Scriptbar
- Diverse Befehle zur Hardware- und Treiberansteuerung
  - Boardinfo
  - Speicher-/Registermanipulation
  - Flash-/Dateisystembefehle



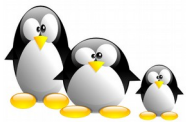
U-Boot Shell

- Grundlegende Befehle
  - help
  - printenv
  - setenv
  - saveenv
- Kommandos für Hardwareansteuerung (etwa fatls oder fatload)
- Erweiterbar mit eigenen Befehlen



U-Boot Standalone Apps

- U-Boot unterstützt dynamisch ladbare ‚Standalone Apps‘
- Apps haben Zugriff auf die U-Boot Befehle
  - Beispielsweise für Hardware-/Systemtests
  - Zusätzliche Bootlogik



Exkurs: GIT

- Verteilte Sourcecodeverwaltung
- Consolenbasiert
- Free/Open Source
- Entwickelt von Linux Torvalds
  (Initiator und Entwickler Linux-Kernel)

https://git-scm.com

Exkurs: GIT

- Features (siehe auch https://de.wikipedia.org/wiki/Git)
  - Nicht-lineare Entwicklung
  - Kein zentraler Server
  - Datentransfer zwischen Repositories
  - Kryptoschutz der History
  - Commit-ID stellt einen SW-Stand dar

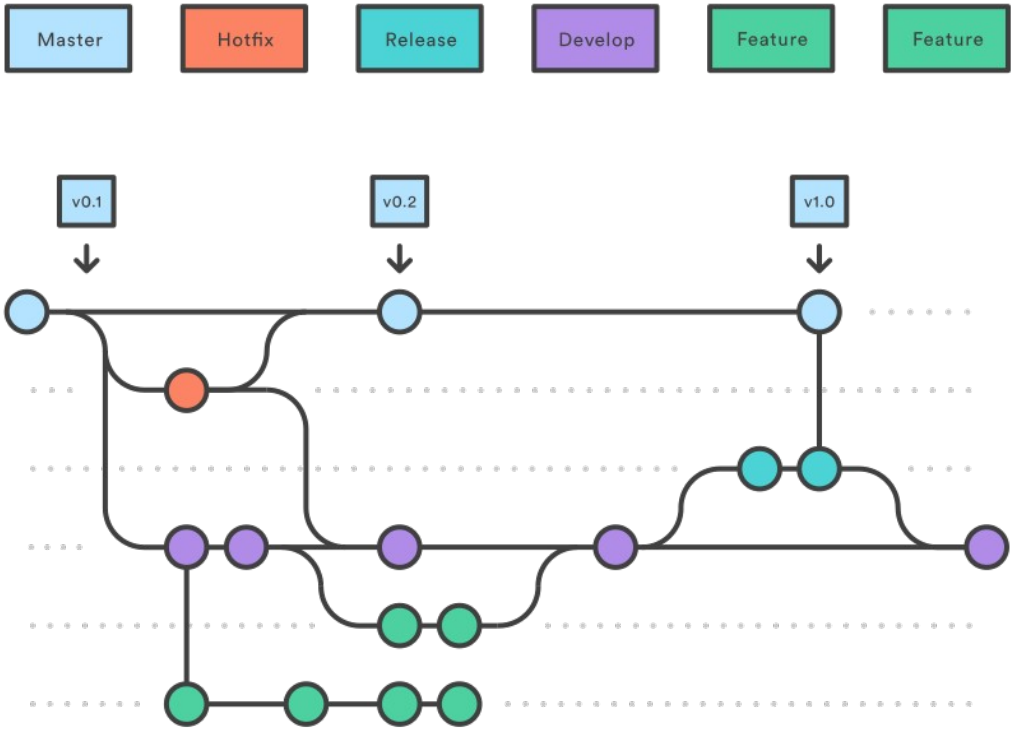
https://git-scm.com

Exkurs: GIT

- Grundlegende Befehle
  - $ git clone <url>
  - $ git push/pull
  - $ git add <file>
  - $ git commit
  - $ git status
  - $ git show <commit-id>
  - $ git log

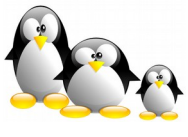
https://git-scm.com

Exkurs: GIT



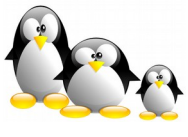
- Branching in Git
  - $ git branch -a
  - $ git checkout <branchname>
  - $ git checkout -b <branchname>
  - $ git merge <branchname>

https://git-scm.com



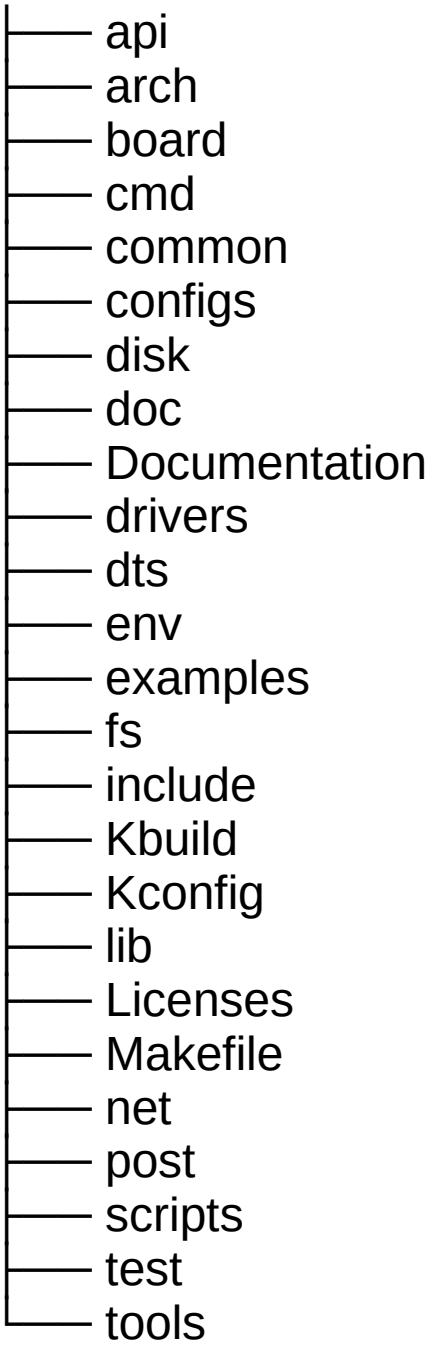
U-Boot Sourcen

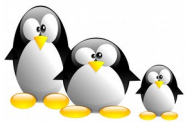
- git://git.denx.de/u-boot.git
- Aktuell: Tag v2019.01
- In Entwicklung: Tag v2019.04



U-Boot Sourcen

- Makefilebasiert
- Konfiguration über Menuconfig
- Board: Boardcode
- Config: Boardkonfiguration
- Include: u. a. weiter Board-beschreibung





Boardcode

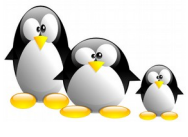
- Beschreibt die Hardware des Boards
  - Speicherinitialisierung
  - Power-Management
  - PIN-Init
  - Peripherieinitialisierung
  - Besonderheiten des Boards

- z. B. sama5d2_xplained.c

Boardconfig

- Devicekonfiguration
  - Welche CPU mit welcher Architektur
  - Welche Peripherie
  - Zusätzliche Software
  - Durch menuconfig erstellt

- z. B. sama5d2_xplained_emmc_defconfig



Boardinclude

- Beschreibung der generischen Teile von U-Boot für das Board
- Konfiguration der Treiber per #define
- Unter anderem
  - Speicherbasisadressen
  - Environment-Settings
  - U-Boot spezifische Settings

U-Boot bauen

- $ export ARCH=arm
- $ export CROSS_COMPILE=arm-linux-gnueabihf-
- $ make sama5d2_xplained_emmc_defconfig
- $ make