



▶ **Marvell.** Moving Forward Faster

Marvell PXA1826 SW

2015



Disclaimer

- ▶ No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document.
- ▶ Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications.
- ▶ With respect to the products described herein, the user or recipient, in the absence of appropriate U.S. government authorization, agrees:
 - 1) Not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2;
 - 2) Not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and,
 - 3) In the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML").
- ▶ At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.
- ▶ Copyright © 1999–2013. Marvell International Ltd. All rights reserved. Alaska, ARMADA, CarrierSpan, Kinoma, Link Street, LinkCrypt, Marvell logo, Marvell, Moving Forward Faster, PISC, Prestera, Qdeo (for chips), QDEO logo (for chips), QuietVideo, Virtual Cable Tester, Xelerated, and Yukon are registered trademarks of Marvell or its affiliates. Avanta, Avastar, DragonFly, HyperDuo, Kirkwood, Marvell Smart, Qdeo, QDEO logo, The World as YOU See It, Vmeta and Wirespeed by Design are trademarks of Marvell or its affiliates.
- ▶ Patent(s) Pending—Products identified in this document may be covered by one or more Marvell patents and/or patent applications.

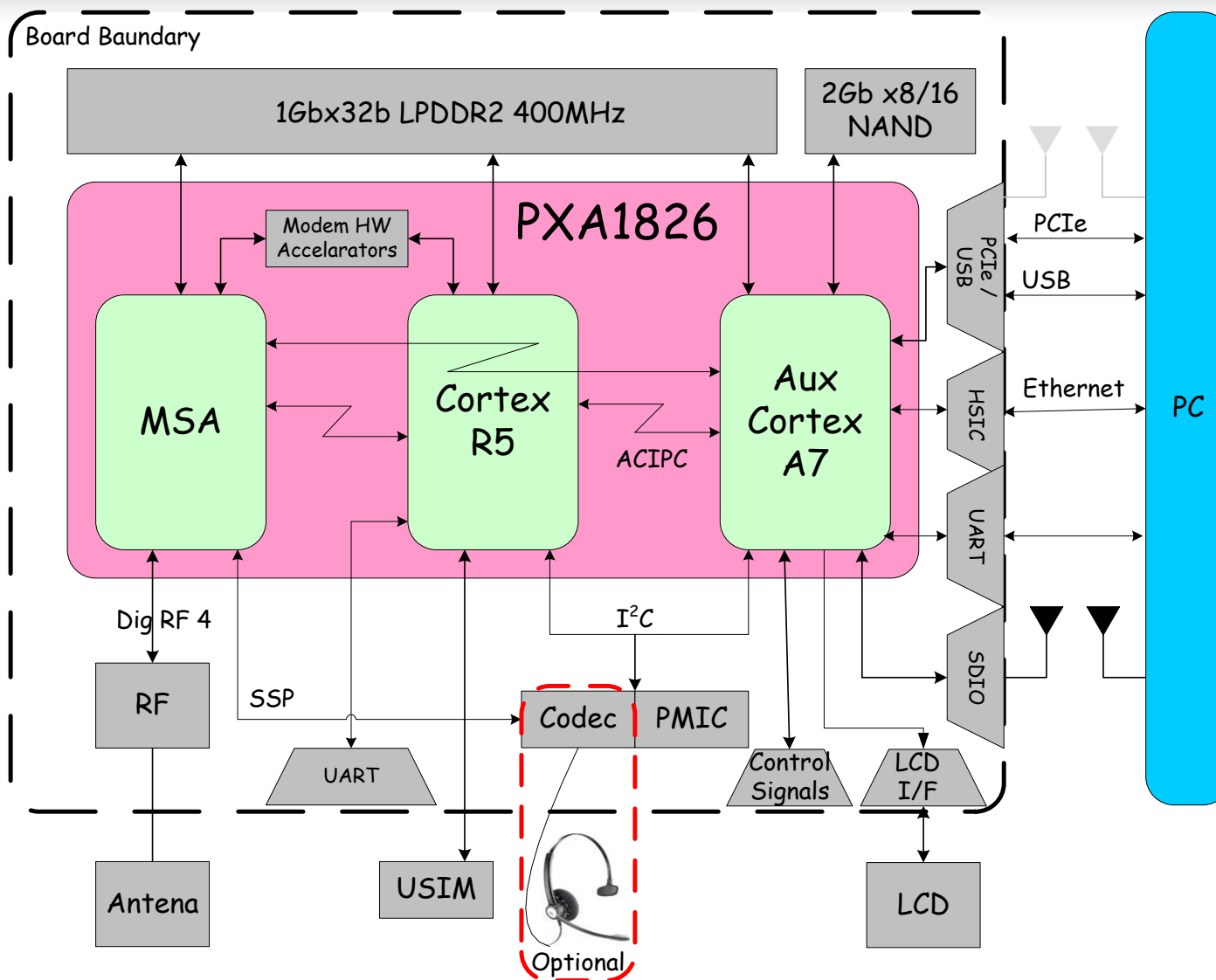
PXA1826 Software Training Agenda (1/2)

- ▶ [General SW overview](#)
- ▶ [Openwrt](#)
- ▶ [OpenWRT Init process](#)
- ▶ [Detailed SW architecture](#)
 - [File system](#)
 - [Memory map](#)
 - [SW Package folder Structure](#)
 - [Telephony Block Diagram](#)
 - [Services](#)
 - [System Configuration](#)
 - [USB](#)
 - [Audio](#)
 - [GPS connectivity Architecture](#)
 - [WiFi, BT](#)

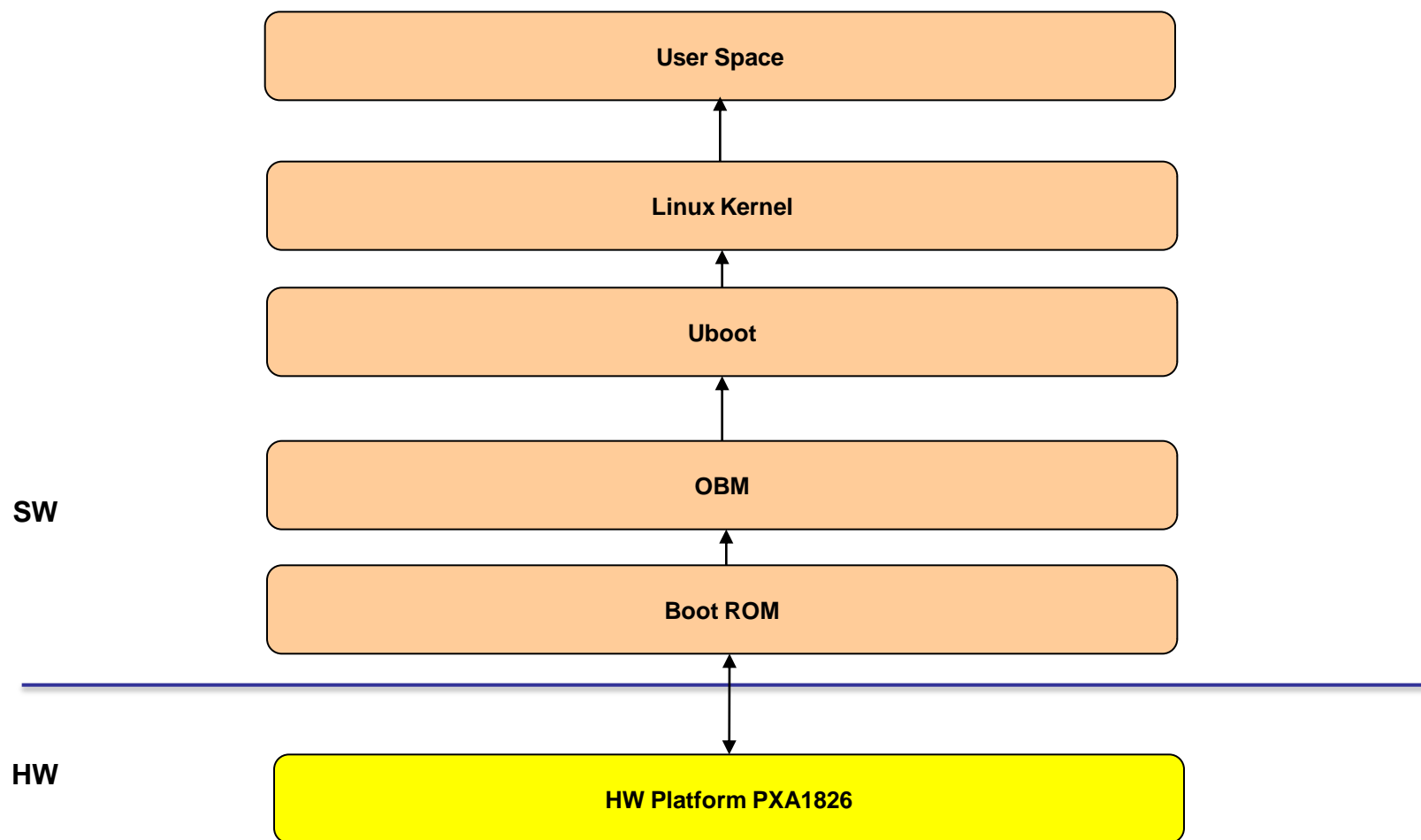
PXA1826 Software Training Agenda (2/2)

- ▶ [Security](#)
- ▶ [NVM](#)
- ▶ [Power Management](#)
- ▶ [Logging \(Diag, EEH, RAMDUMP\)](#)
- ▶ [FOTA](#)
- ▶ [SW Porting](#) & **Adding a new profile**

PXA1826 Components and Interfaces



PXA1826 SW Components



PXA1826 SW Components

▶ **BootROM**

- Check signature (TIM)
- **DDR**, Flash configuration
- Load OBM

▶ **OBM**

- Load UBOOT, obm2osl
- Security check for relevant images (Uboot, Kernel)

▶ **UBOOT**

- Load zImage
- **Platform configurations**

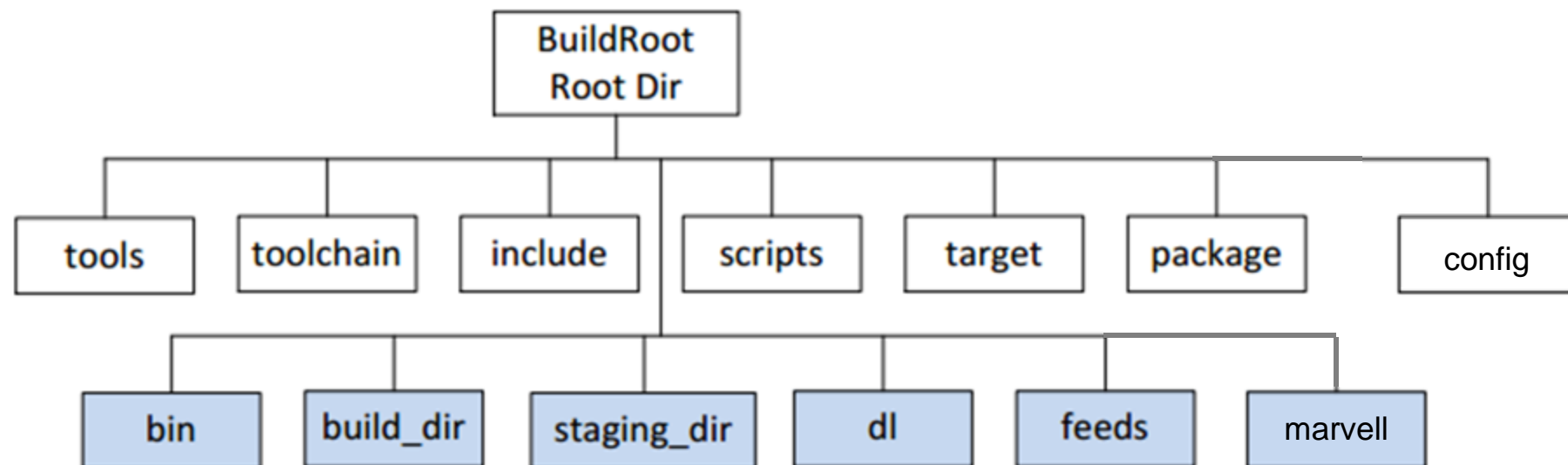
▶ **Kernel 3.10**

- Android compatible Kernel
- Device tree

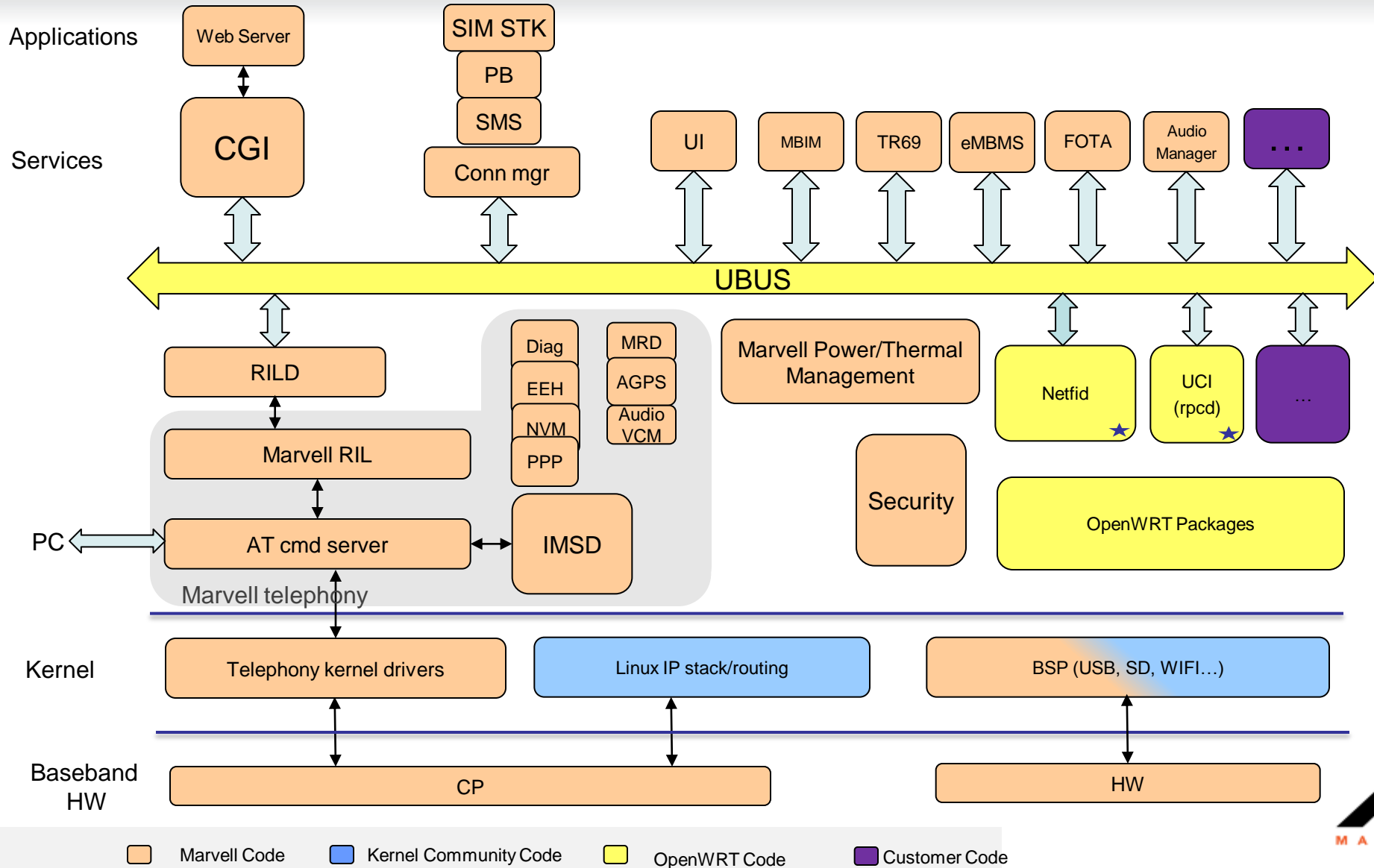
▶ **OpenWRT**

SW Package folder Structure

- OpenWRT is becoming “android” of gateway
- Very useful for low memory solutions
- **OpenWRT is very scalable and flexible:**
 - one config file for all SW layers; Kernel, User Space etc.
 - Package methodology
 - Marvell code is united under dedicated folder



PXA1826 SW Architecture (Cont.)





OpenWRT

OpenWrt
Wireless Freedom

OpenWRT

A free and open Linux based SW (FW) for embedded devices (mainly routers)

- Version: “Barrier Breaker”
- Named “WRT” after the first device that used Linux on it “Linksys WRT45G”
- First release: Jan 2004
- OpenWRT is becoming “android” of gateway
- OpenWRT is very scalable and flexible:
 - Very well designed package methodology allows very easy customizations and managements
 - one config file for all SW layers; Kernel, User Space etc
 - Support independent APP developments, allow [.IPK](#) installation and management
 - Strong community supports around the world, thousands of APPs



OpenWRT Environment

- ▶ OpenWrt Buildroot environment is a collection of Makefiles, patches and scripts, which generates the cross-compilation toolchain, downloads Linux kernel, generates a root file system, manages 3rd party packages, etc.
- ▶ The cross-compilation toolchain uses [uClibc](#)
- ▶ In the OpenWrt Buildroot source tree, there is no Linux kernel or any source code tarballs of the 3rd party packages. The collection of Makefiles determines the version of Linux kernel to download, and the version of the package tarball to download and compiled in to the image.

OpenWRT Buildroot Source Tree

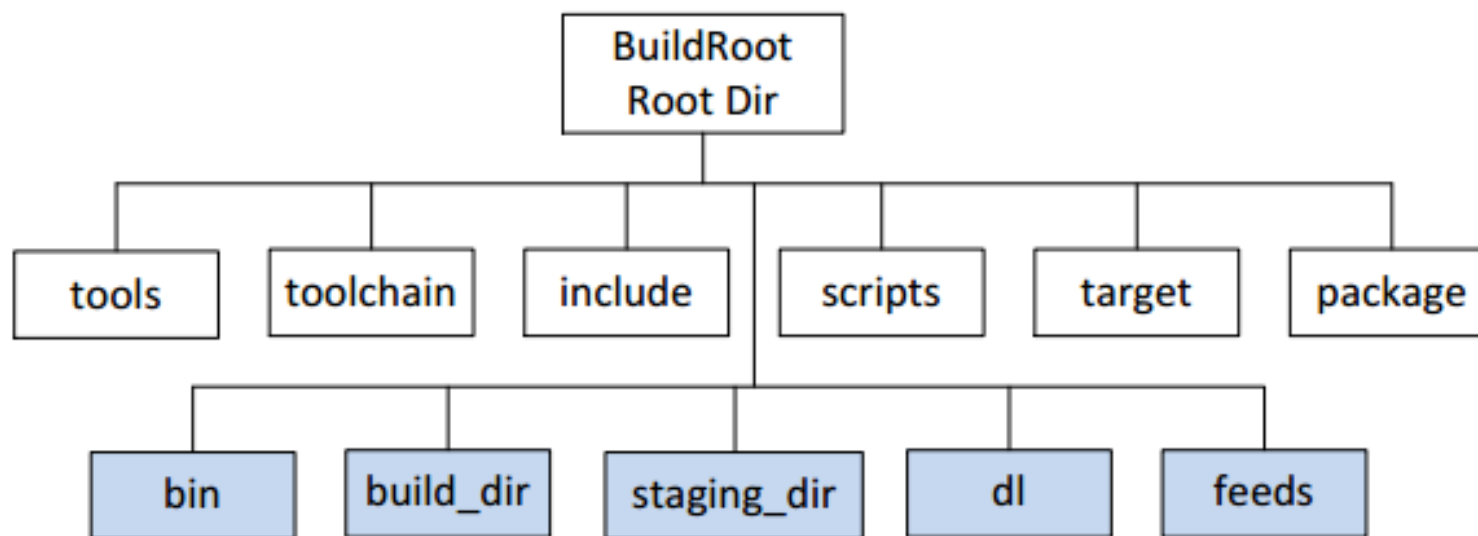


Figure 1. OpenWrt Buildroot Source Tree (The folders in the second lines are generated during compilation).

OpenWRT Buildroot Source Tree

- ▶ **tools** – contains all the build instructions to fetch the image **building tools**
- ▶ **toolchain** - contains all the build instructions to fetch the **kernel headers, the C library, the bin-utils, the compiler itself and the debugger**. If you add a completely new architecture, you would add a configuration for the C library here.
- ▶ **target** - build instruction for firmware image generating process and for the kernel building process; compiles kernel and firmware image utilities, builds firmware image, generate Image Generator (former called Image Builder)
- ▶ **package** – the OpenWrt Makefiles and patches for all the main packages. **The OpenWrt Makefile has its own syntax, different from the conventional Makefile of Linux make tool.** The OpenWrt Make file defines the meta information of the package, where to download the package, how to compile, where to installed the compiled binaries, etc.
- ▶ **include**
- ▶ **scripts** – sh and perl scripts for the OpenWrt package management

OpenWRT Buildroot Source Tree

- ▶ **dl** –(down link) Where the package tarballs will be downloaded
- ▶ **build_dir** – where all tools will be cross-compiled
- ▶ **staging_dir** – where the cross-compilation tools will be installed
- ▶ **feeds** – OpenWrt packages management for packages that are not in the openwrt basic structure.
 - Possible to create your own package
- ▶ **bin** – where the firmware image will be generated and all the .ipk package files will be generated

Image building steps (once the configuration is done)

1. Download the cross-compilation tools, kernel headers, etc. and
2. Set up the staging directory (**staging_dir /**). This is where the cross-compilation toolchain will be installed. If you want to use the same cross-compilation toolchain for other purposes, such as compiling third-party applications, you can find the cross-compiler tools in this directory, and then use `arch-linux-gcc` to compile your application.
3. Create the download directory (**dl/** by default). This is where the tarballs will be downloaded.
4. Create the build directory (**build_dir/**). This is where all user-space tools will be **compiled**.
5. Create the target directory (**build_dir/target-arch/root** by default) and the target filesystem skeleton. This directory will contain the final root filesystem.
6. **Install** the user-space packages to the root file system and compress the whole root file system with proper format. The result firmware image is generated in **bin/**

OpenWRT Compilation options

▶ Download package from network

- Package will be downloaded from network in the compilation process (if not exist)
- Compile in build directory
- Makefile example: network/services/dropbear/Makefile

▶ Compile from source directory

- In our system, Kernel & Telephony are compiled like that
- Compile in source directory (recommended)
 - Makefile example: libs/libprop2uci/Makefile
- Compile in build directory
 - Makefile example: network/services/lte-telephony/Makefile

▶ Changing downloaded package code

- In order to change code of downloaded package, we add patches that during compilation time are added to the unzipped code.

UBUS Concept

- ▶ To provide communication between various daemons and applications
- ▶ It consists of few parts including daemon, library and some extra helpers.
- ▶ The heart of this project is ubusd daemon. It provides interface for other daemons to register themselves as well as sending messages.
- ▶ This interface is implemented using **Unix socket** and it uses **TLV (type-length-value) messages**.
- ▶ To simplify development of software using ubus (connecting to it) a library called libubus has been created.
- ▶ **Command-line ubus tool**
 - The ubus command line tool allows to interact with the ubusd server (with all currently registered services).
 - It's useful for investigating/debugging registered namespaces as well as writing shell scripts.
 - For calling procedures with parameters and returning responses it uses user-friendly JSON format.



OpenWRT Init process

► procd

- When Kernel boot is finished, procd is called.
- procd is OpenWrt [process](#) management daemon. It keeps track of processes started from init scripts (via ubus calls)
- Boot stages:
- procd: preinit
 - Early mount: proc, sysfs, tmpfs (DDR FS)
 - Creating needed devices (block, character)
 - Console is alive
 - Calls preinit and system functions
 - Calls inittab – linux script
 - that selects the initial script according to the start grade
 - production mode modification
 - Copy All init scripts from init.d to rc.d
 - Run hotplugs
 - UBUS init

OpenWRT Init process

- procd: - init –
 - Run all scripts located in : located /etc/rc.d according to the grades
 - Example: build_dir/target-arm_cortex-a7+neon-vfpv4_uClibc-0.9.33.2_eabi/root-mmp/etc/init.d/mrvl_init

<http://wiki.openwrt.org/doc/techref/process.boot>

Initramfs

- ▶ This is used for actions needed to be performed before the root partition is mounted.
- ▶ Useful if you need to do something special to get your root partition visible to the kernel.
- ▶ InitramFS is currently not used in our system, but openWRT supports it.

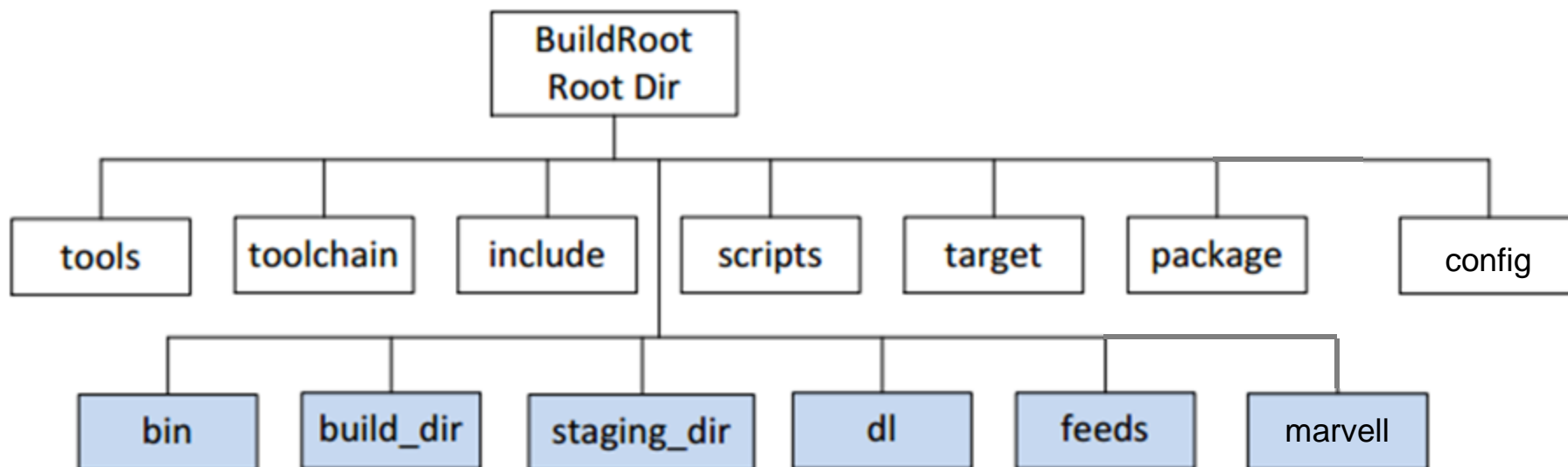




Detailed SW Architecture

SW Package folder Structure

- ▶ Fully aligned to openwrt base structure
- ▶ marvell folder contains all Marvell specific modules and services (obm, uboot, kernel, telephony)
- ▶ Config folder contains platform configuration files



File System

▶ Currently:

- Read only FS: squashfs (version 4.0)
- OverlayFS, NVM: JFFS2 (version 2.2)

▶ Supports UBIFS by configuration



Memory MAP - NAND

Description	Partition #	Partition Name	Flash offset	Partition End	Decimal offset	Size (MB)	Size	Size Hex	Blocks	Format	Access permission	Image Name
TIM	0	BootLoader	0	2000	0	0.01	8,192	2000	1	Raw		
OBM	0	BootLoader	2000	20000	8192	0.12	122,880	1E000	1	Raw		pxa1826_NonTLoader_NAND_DDR.bin
MRD	1	Reliabledata	20000	40000	131,072	0.13	131,072	20000	1	Raw	Running: RO, Factory: WR	ReliableData.bin
MRD Backup	2	Reliabledata	40000	60000	262,144	0.13	131,072	20000	1	Raw	Running: RO, Factory: WR	
MEP2	3	MEP2	60000	80000	393,216	0.13	131,072	20000	1	Raw	Running: RO, Recovery: RW	
secure store	3	MEP2	80000	A0000	524,288	0.13	131,072	20000	1	Raw	Running: RO, Recovery: RW	
DTIM	4	dtim	A0000	E0000	655,360	0.25	262,144	40000	2	Raw	Running: RO, Recovery: RW	
RF Plugin	5	LWG	E0000	120000	917,504	0.25	262,144	40000	2	Raw	Running: RO, Recovery: RW	
MSA	5	LWG	120000	420000	1,179,648	3.00	3,145,728	300000	24	Raw	Running: RO, Recovery: RW	
Arbel	5	LWG	420000	E20000	4,325,376	10.00	10,485,760	A00000	80	Raw	Running: RO, Recovery: RW	
RF Plugin	6	LTG	E20000	E60000	14,811,136	0.25	262,144	40000	2	Raw	Running: RO, Recovery: RW	
MSA	6	LTG	E60000	1160000	15,073,280	3.00	3,145,728	300000	24	Raw	Running: RO, Recovery: RW	
Arbel	6	LTG	1160000	1B60000	18,219,008	10.00	10,485,760	A00000	80	Raw	Running: RO, Recovery: RW	
NVM	7	NVM	1B60000	1E60000	28,704,768	3.00	3,145,728	300000	24	ubifs	Running: RW, Recovery: N/A	openwrt-mmp-pxa1826-nvm.ubifs
uboot and env	8	uboot	1E60000	1EE0000	31,850,496	0.50	524,288	80000	4	Raw	Running: RO, Recovery: RW	uboot.bin
zImage	9	kernel	1EE0000	22E0000	32,374,784	4.00	4,194,304	400000	32	Raw	Running: RO, Recovery: RW	zImage
rootfs	10	rootfs	22E0000	32E0000	36,569,088	16.00	16,777,216	1000000	128	squashfs	Running: RO, Recovery: RW	Ramdisk.img
rootfs data	10	rootfs	32E0000	4E80000	53,346,304	27.63	28,966,912	1BA0000	221	ubifs	Running: RO, Recovery: RW	sqashfs
mdb_file	11	MDB	4E80000	5C80000	82,313,216	14.00	14,680,064	E00000	112	squashfs	Running: RW, Recovery: N/A	mdb.ubifs
mass_storage	12	massstorage	5C80000	5D80000	96,993,280	1.00	1,048,576	100000	8	vFAT	Running: RW, Recovery: RW	
OTA reserved	13	OTA	5D80000	7D80000	98,041,856	32.00	33,554,432	2000000	256	raw	Running: RW, Recovery: RW	
MRVL BBM		MRVL_BBM	7D80000	8000000	131,596,288	2.50	2,621,440	280000	20		Auto generated	
End of flash			8000000	8000000	134,217,728	0.00	0	0				
				Flash Size (dec)	134,217,728							
				map status	OK							

Memory MAP - DDR

Type	Size	Used Area	Load OBM	Load Uboot	Load Kernel	Extact Kernel	Init Kernel	System Start	Comments
SRAM	64KB	0xD1000000	TIM						0xD1000000
			Bootrom						
DDR 128M		0x00000000		ATAG					
		0x00004000					MMU-Table	MMU-Table	
		0x00007FC0							
	~7MB	0x00008000			zImage	vmlinux		vmlinux	current NAND version __log_buf@0xc05c1018, SPI-NOR __log_buf@0xc047d018.
		0x003FFFFFFF							
		0x004000000		obm2osl					
		0x00407FFF							
		0x00408000		Uboot					
		0x0049FFFF							
		0x004E0000	OBM						
		0x00580000							
		0x007FFFFFFF							
		0x009FFFFFFF						Kernel usable	
	4KB	0x00A00400						Crash kernel	ramdump signature.
		0x00A013FF							
	77MB+1020K	0x00A01000						Kernel usable	
		0x057FFFFFFF							
	40MB	0x05800000						CP Reserved	Loaded by cloader kernel module
		0x05900000							
		0x05C00000							
		0x07FFFFFFF							

BLF file

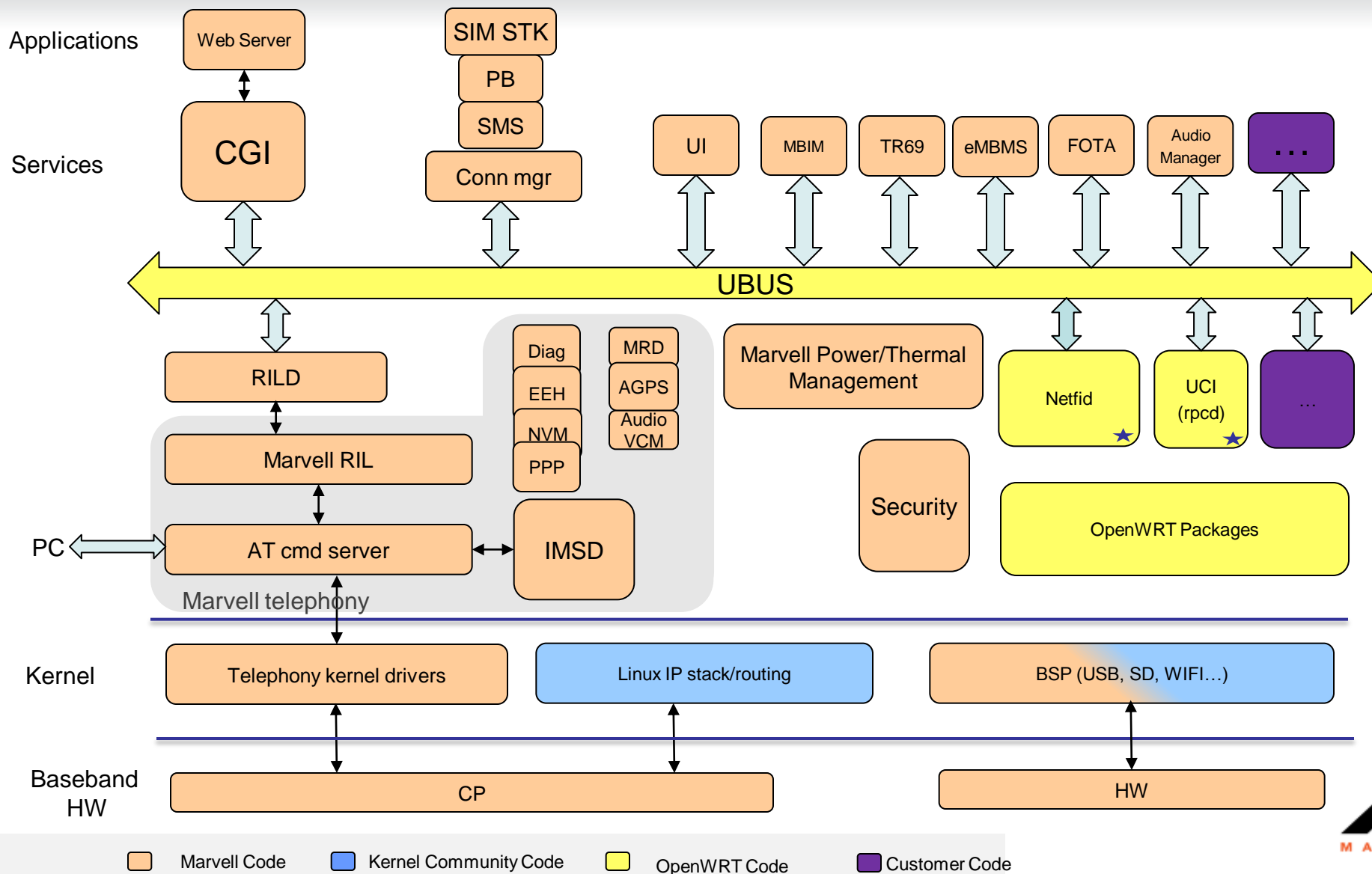
S..	P...	T...	ImgID	Erase Size	Flash Address	Load Address	HashAlgm	Size to Hash	ImgType	Img File Path
<input checked="" type="checkbox"/>	0	1	TIMH		0x00000000	0xD100_0000	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\tim_nezha3.bin
<input checked="" type="checkbox"/>	0	1	OBTMI		0x00001000	0x0000_0000	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\pxa1826_TLoader_NAND.bin
<input checked="" type="checkbox"/>	0	0	RBLI		0x00020000	0xFFFF_FFFF			RAW	C:\temp\swd\pxa1826_ReliableData.bin
<input checked="" type="checkbox"/>	0	2	ARBI		0x00420000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\pxa1826_ARBEL_LWG.bin
<input checked="" type="checkbox"/>	0	2	GRBI		0x00120000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\pxa1826_MSA_LWG.bin
<input checked="" type="checkbox"/>	0	2	RFBI		0x000E0000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\pxa1826_RFPLUGIN_CA_ATT_LWG.bin
<input checked="" type="checkbox"/>	0	2	ARB2		0x01160000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\pxa1826_ARBEL_LTG.bin
<input checked="" type="checkbox"/>	0	2	GRB2		0x00E60000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\pxa1826_MSA_LTG.bin
<input checked="" type="checkbox"/>	0	2	RFB2		0x00E20000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\pxa1826_RFPLUGIN_CA_ATT_LTG.bin
<input checked="" type="checkbox"/>	0	2	OSLO		0x01E60000	0x0050_0000	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\openwrt-mmp-pxa1826-u-boot.bin
<input checked="" type="checkbox"/>	0	2	ZIMG		0x01EE0000	0x0010_8000	SHA-256	0xFFFF_FFFF	RAW	C:\temp\swd\openwrt-mmp-pxa1826-zImage
<input checked="" type="checkbox"/>	0	0	SYSJ	0x034A_0000	0x022E0000	0xFFFF_FFFF			RAW	C:\temp\swd\openwrt-mmp-pxa1826-root.squashfs
<input checked="" type="checkbox"/>	0	0	MDBI		0x05780000	0xFFFF_FFFF			RAW	C:\temp\swd\pxa1826_mdb.squashfs
<input checked="" type="checkbox"/>	0	2	TIM1		0x00080000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	DTIM	C:\temp\swd\DTim.Primary
<input checked="" type="checkbox"/>	0	6	TIM4		0x000C0000	0xFFFF_FFFF	SHA-256	0xFFFF_FFFF	DTIM	C:\temp\swd\DTim.PPsetting

MTD

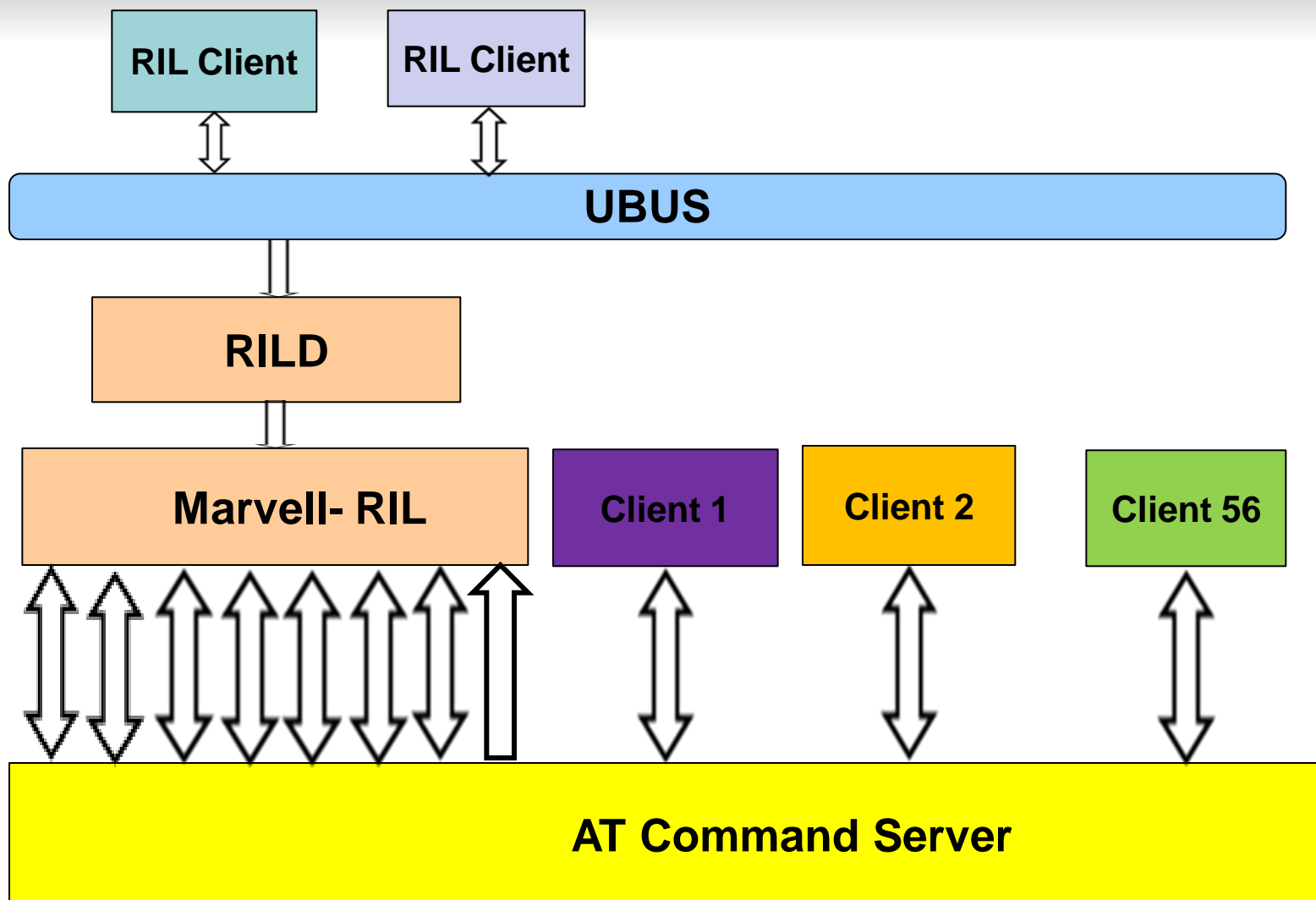
- ▶ [1.654357] 0x000000000000-0x000000020000 : "bootloader"
- ▶ [1.660736] 0x000000020000-0x000000040000 : "reliabledata"
- ▶ [1.667144] 0x000000040000-0x000000060000 : "reliabledata2"
- ▶ [1.673767] 0x000000060000-0x0000000a0000 : "mep2"
- ▶ [1.679595] 0x0000000a0000-0x0000000e0000 : "dtim"
- ▶ [1.685394] 0x0000000e0000-0x0000001d60000 : "cpimage"
- ▶ [1.691558] 0x0000001d60000-0x0000001e60000 : "NVM"
- ▶ [1.697174] 0x0000001e60000-0x0000001ee0000 : "u-boot"
- ▶ [1.703155] 0x0000001ee0000-0x00000022e0000 : "kernel"
- ▶ [1.709167] 0x00000022e0000-0x0000005780000 : "rootfs"
- ▶ [1.715118] mtd: device 9 (rootfs) set to be root filesystem
- ▶ [1.721160] mtd: partition "rootfs_data" created automatically, ofs=0x32e0000, len=0x24a0000
- ▶ [1.729675] 0x00000032e0000-0x0000005780000 : "rootfs_data"
- ▶ [1.736083] 0x0000005780000-0x0000005c80000 : "mdb"
- ▶ [1.741943] 0x0000005c80000-0x0000005d80000 : "misc"
- ▶ [1.747711] 0x0000005d80000-0x0000007d80000 : "OTA"



PXA1826 SW Architecture (Cont.)



PXA1826 Telephony Block Diagram



services

▶ Phonebook

- Service that handles contact storage on SIM
 - Allow UBUS interface for saving/pulling numbers from SIM
 - UI uses it to display the contacts

▶ Sim manager

- Service that implement Sim related functionalities:
 - Sim status request
 - Enable/enable/change pin
 - reset_pin_using_puk

▶ SMS service

- Service that implement SMS related functionalities
- Currently support PDU only
- Option to list, save, send, delete, move, query SMS.

services (cont.)

▶ **traffic_stat:**

- Statistics
- Data limitation

▶ **Wireless**

- Wifi configuration
- Mainly used for UI

▶ **router_firewall , router_settings**

- Rerouting and firewall configuration over UBUS (instead of changing UCI file) – used for UI

▶ **netmode_mgr**

- Webui for setting and getting network operations (getting cells etc)

▶ **mgui/**

- GUI for LCD

▶ **Charger**

Services (cont.)

- ▶ **MBIM** - Mobile broadband (MB) Identity Morphing. The solution maps the morphing device's USB configuration to a set of USB functions. At any point in time, a single set of functions (by way of a configuration) are exposed to the host. eliminates the need for distributing the driver package
- ▶ **LWM2M** - Lightweight M2M, a protocol from the Open Mobile Alliance for [M2M](#).
 - Provide Device Management functionality over cellular networks
 - Transfer service data from the network to devices
 - Extend to meet the requirements of most any application
- ▶ **OMA-DM** – will be supported after LWM2M.

Services (cont.)

- ▶ **TR69** - (Technical Report 069) is a Broad band forum (formerly known as DSL forum). Automatic configuration and management of modems, routers, gateways by Auto Configuration Servers (ACS)
- ▶ **eMBMS** - Multimedia Broadcast Multicast Services (MBMS) is a point-to-multipoint interface. Target applications include mobile TV and radio broadcasting.
- ▶ **Connection Manager**



System Configuration

- ▶ Pxa1826 configuration is set at compilation and the result is BLF file.
- ▶ The flags are added to DTIM
- ▶ OBM reads the configuration and transfer it to UBOOT
- ▶ UBOOT handle accordingly and notify Kernel by as a cmdline argument
- ▶ Some configuration are transpose to UCI file

- ▶ Example: NZA3/cfg_files/pxa1826/ext_blf_cfg_PROD

- ▶ possible to change this configuration in runtime (changing flags in flash or cmdline in UBOOT)

System Configuration (cont.)

- ▶ Current configuration:
 - PIPE (0 - router, 1 - pipe mode)
 - IMSD (with or without IMS)
 - EEHP(stall or Ramdump)
 - PROD (0 – normal mode, 1 – production mode)
- ▶ Plan to support audio, GPS, WiFi and other configuration (same as pxa1801)
- ▶ Example:
 - IMS (BLF, cmdline, UCI, services.init)
 - Prod (BLF, UBOOT, cmdline, UCI)

Android wrappers

- ▶ ml_utilset
 - Android utilities wrappers
- ▶ prop2uci
 - Wrappers for getprop, setprop using UCI
- location:
 - marvell\services\android_wrapper\

System data configuration

- ▶ Router Mode:
 - fastpath v2
 - Connection manager
 - Data path
 - Data optimization
 - WebUI (in MiFi only)

System data configuration

▶ Pipe Mode :

- No fastpath (pipe module instead)
- Pipe demon
- No WebuUI
- CM on target for 1 PDP
- CM application on host FOR Multiple PDPs (need to implement)



USB

- ▶ Two controllers sharing the same PHY:
 - USB2 – use the same USB controller as pxa1801 (silicon image IP)
 - USB3 (also has USB2 support built-in) – we use only the part of USB3.
- ▶ For OTG and USB2 we still use USB2 controller.
- ▶ Frame-work – since our Kernel is Android based, **we use Android USB frame-work.**

USB Configuration:

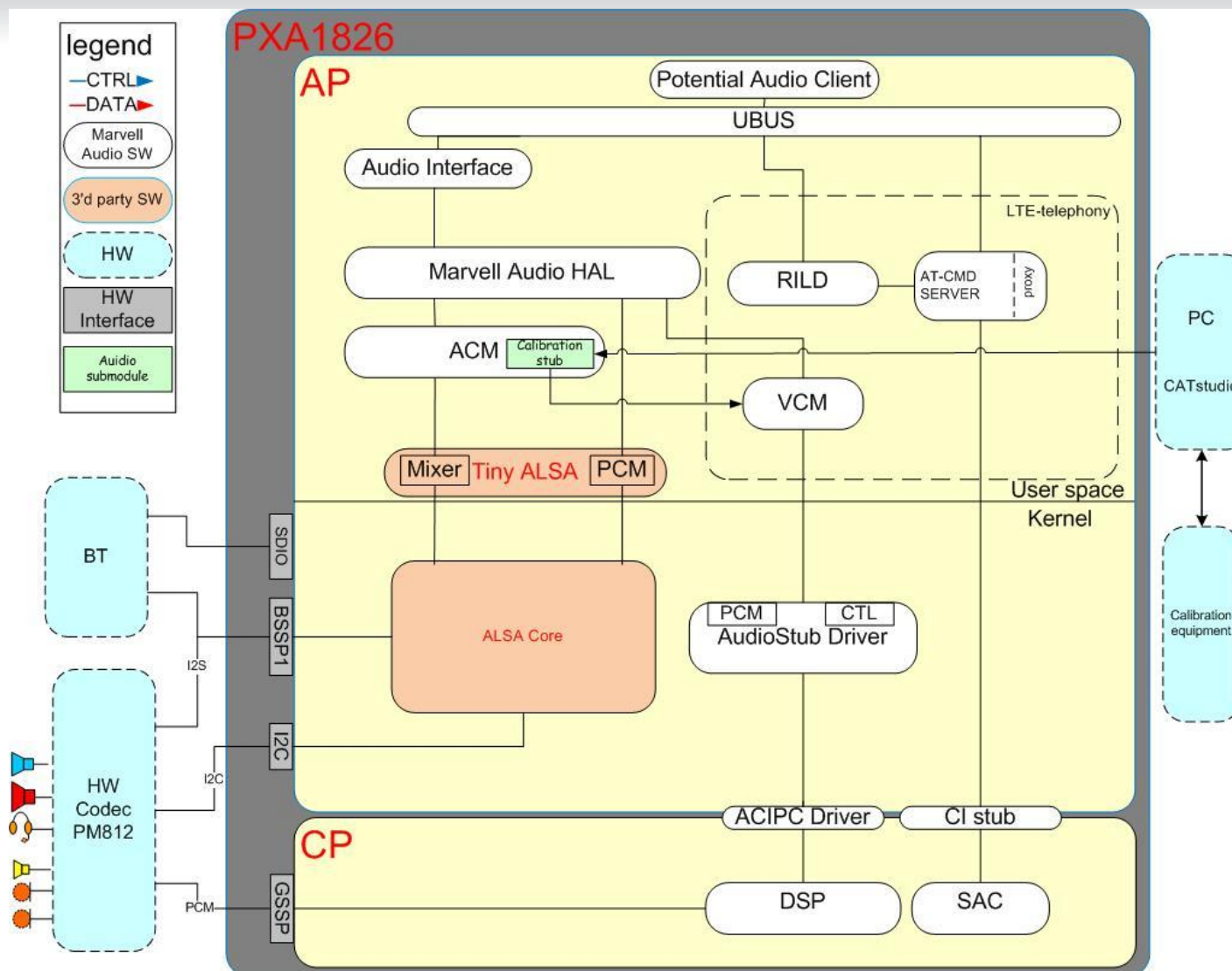
- ▶ USB 2/3 detection – start as USB3 and fallback to USB2.
- ▶ OTG detection – role switch is static according to cable ID.

USB

- ▶ OS detection (works with standard drivers)
 - IOS – loading **ECM**
 - Linux*, Win 7 – loading **RNDIS**
 - Win 8 and up – loading **MBIM**
 - In addition, we support the following function by default:
 - marvell_diag
 - marvell_modem
 - Sulog
 - We can support also mass_storage, ncm_function on demand
- ▶ **For USB3 - there is end-point limitation so we can only support up to four functions.**
- ▶ **Linux native RNDIS HOST driver support multiple packets up to 2K.**
 - Causes reduction in throughput.
 - Throughput can be increased by applying a patch on the HOST (achieve same throughput as windows)
 - In any case, the detection of max transfer size is automatic by SW (it will work also without the patch just with lower throughput).
- ▶ **usbfs is supported but currently not used.**



Audio Architecture



Standard/Common audio functionality

Standard/Common audio functionality is handled via audioHAL, while triggered by either AT CMD and/or UBUS application (e.g. IMS)

- Examples:
 - Start/stop voice call
 - Set audio profile (SPKR, HS, etc')
 - Set volume/mute
 - More (HiFi, BT, etc')
- Since the audio solution is based on tinyALSA (and not include ALSA amixer), AT CMDs will need to be extended. All pxa1801 functionalities will be supported through ATCMD
- **Flexibility: CODEC control using XML files.** (easy to switch CODEC)

PCM data streaming

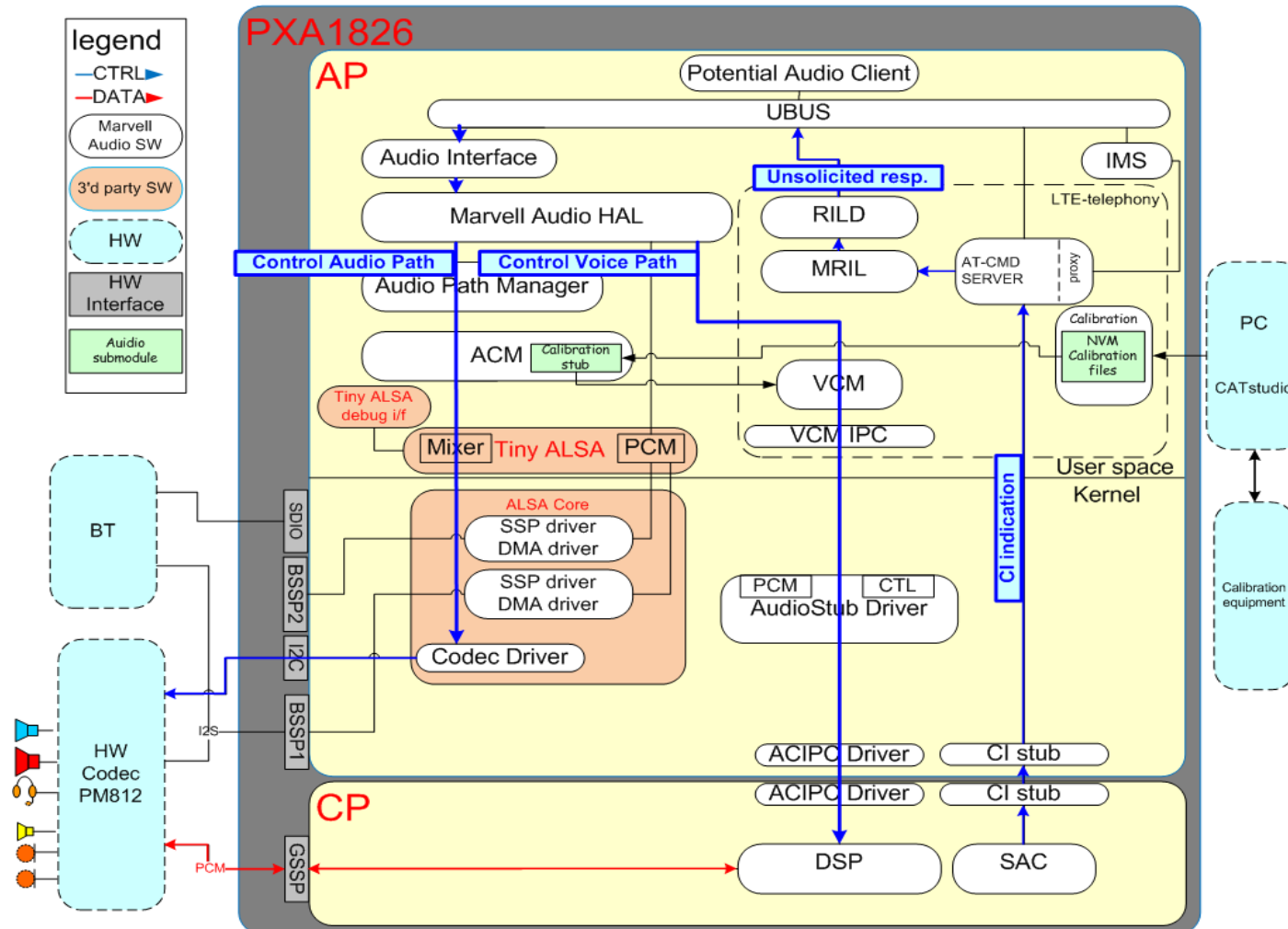
3 bi-directional (playback/record) audio PCM data streaming are available.

Support 2 simultaneously:

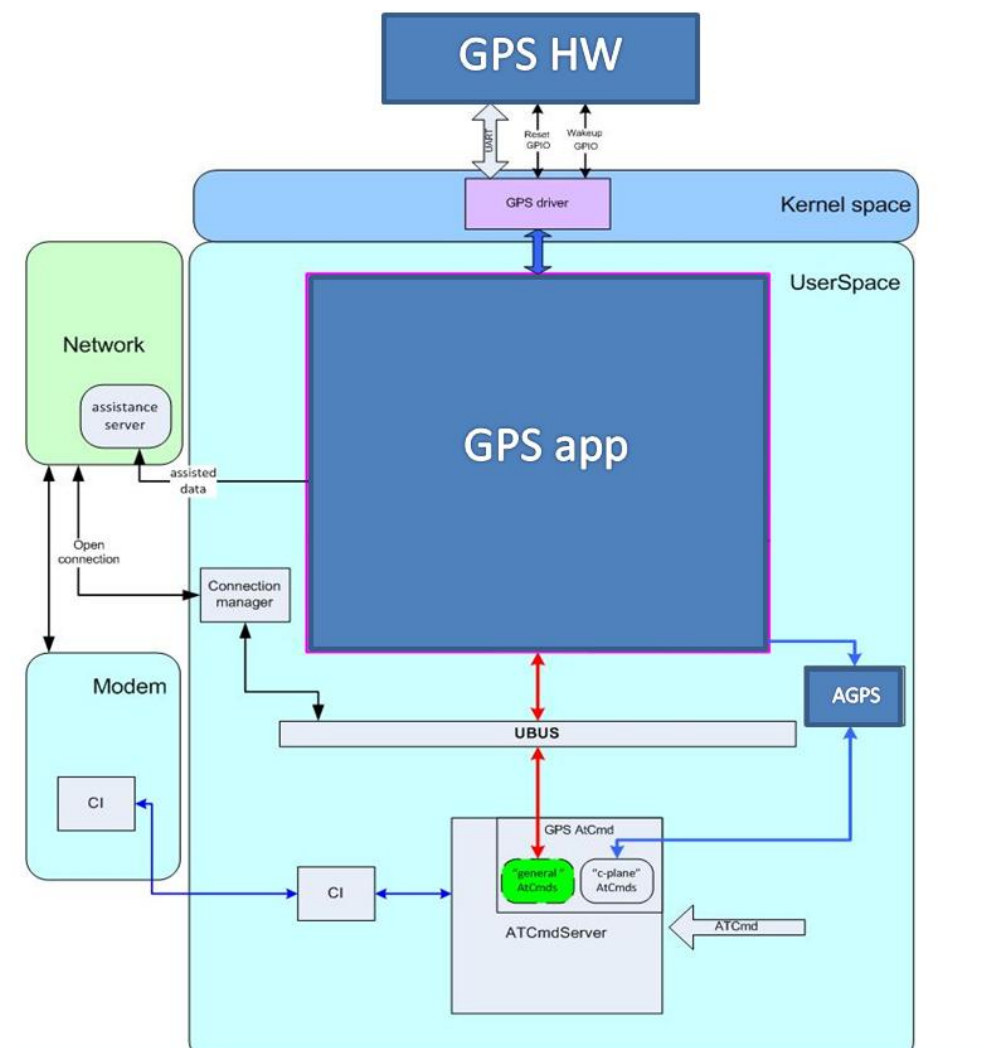
1. Apps audio data #1: 8-48 kHz , via BSSP1
 2. Apps audio data #2: 8-48 kHz, via BSSP2
 3. Comm audio data : 8,16 kHz (WB) , via GSSP (PCM streaming from Apps is also supported).
- Each interface, physically, can be configured as I2S,PCM, etc'.

Standard/Common audio functionality

Start VC



GPS connectivity Architecture



WiFi + BT

- ▶ Support for the below Chips:
 - 8897 – 802.11ac/n/a/g/b 2x2, BT 4.0+HS, NFC (default)
 - 8887 – 802.11ac/n/a/g/b 1x1, BT 4.0, FM
 - 8797 – 802.11n/a/g/b 2x2, BT 4.0+HS, FM (Same as in Omer2)
 - 8787 – 802.11n/a/g/b 1x1, BT 3.0+HS, FM
 - 8777 - 802.11n/g/b 1x1, BT 4.0, FM
- ▶ Automatic detection + drivers uploading
- ▶ Supports AP and AP + STA modes (hotspot).
- ▶ BT service – carnally support blueZ
- ▶ The following profiles would be supported in the future:
 - HFP
 - PBAP A2DP
 - AVRCP
 - DUN
 - OPP



Security in PXA1826

- ▶ Security is the concept of maintaining the root of trust (RoT)
- ▶ It consist of few entities
 - Trusted Boot: the first step of establishing the RoT
 - Marvell provided security libraries: implementing security standards and are checked at boot .
- ▶ **PXA1826 SoC Security Building Blocks**
 - BootROM
 - GEU – Generic Encryption and Fuse Unit
 - Fuse, Life Cycle State (LCS)
- ▶ **PXA1826 Security Solutions**
 - WTPSP – library that use users pace security related functions to GEU
 - Trusted Boot
 - Secure OTA
 - Secure MRD

MRD - Marvell Reliable Data

- ▶ MRD is a data area in flash
- ▶ The MRD data is encrypted and integrity protected.
- ▶ MRD content is read from flash, integrity-checked and decrypted and then copied to the DDR on each boot.
- ▶ MRD contains target-specific (unique and non-unique) sensitive data.

MRD content:

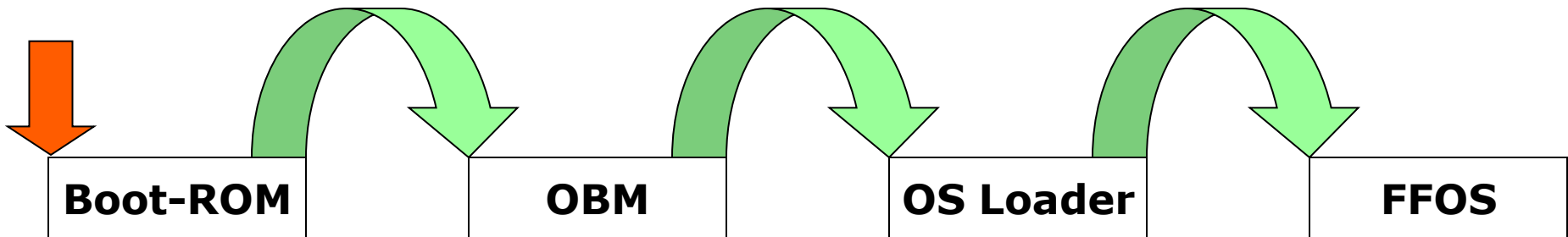
- RF calibration results
- Serial Number - Optional
- IMEI
- MEP (SIM Lock) data – Optional
- WiFi MAC address - Optional
- BT-ID - Optional

PXA1826 Trusted boot overview

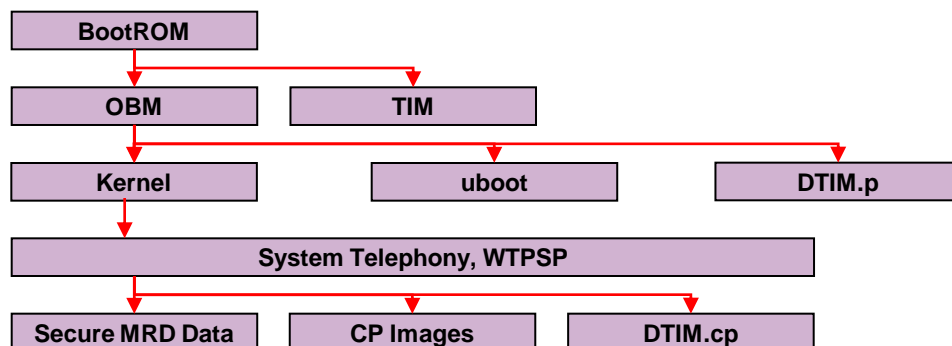
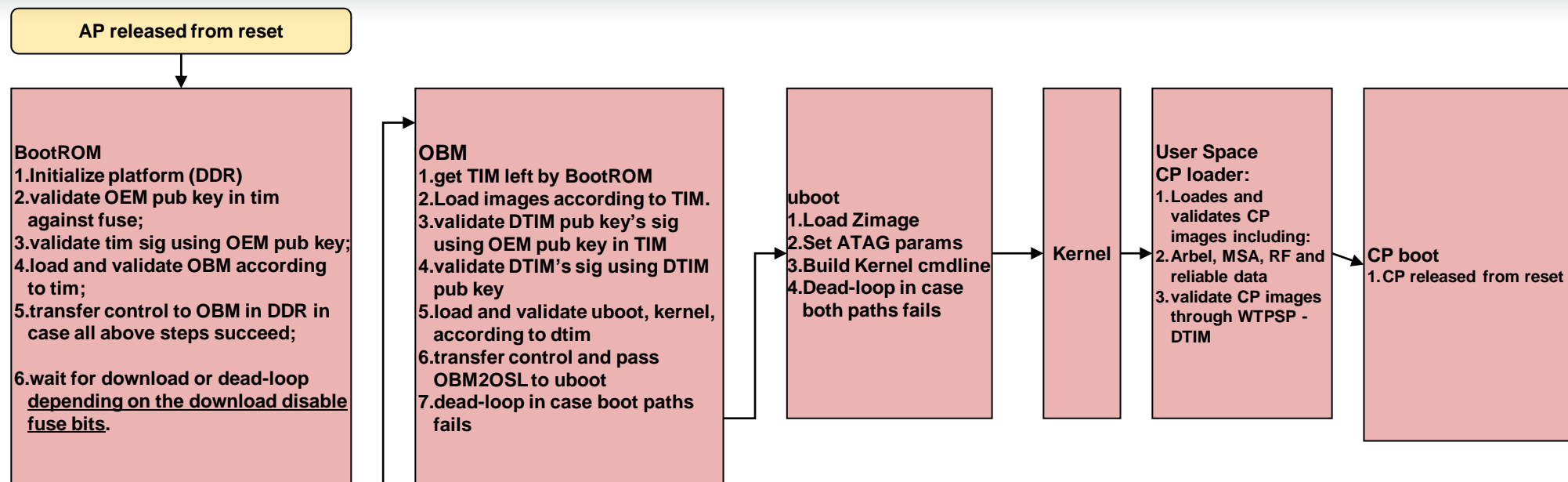
- ▶ Trusted boot is a package of hardware and software components that allow an OEM to perform security validation on a platform at boot time.
- ▶ Boot ROM uses SHA1 hash and the Digital Signature of images stored in flash to establish the root of trust.
- ▶ The security parameters are stored in flash using a data structure called TIM.
- ▶ Validation always starts with the BootROM.
- ▶ A RSA asymmetric key is required and also secured provisioning of this key into one-time programmable fuses before the device is deployed.

PXA1826 Trusted Boot Flow

- ▶ Boot process starts with reset and ends with FFOS upload and running
- ▶ Main functions:
 - load **validated** code from flash to DDR
 - configure platform HW prior to loading OS



Detailed CA7-Core0 Trusted Boot Flow and Trust Chain

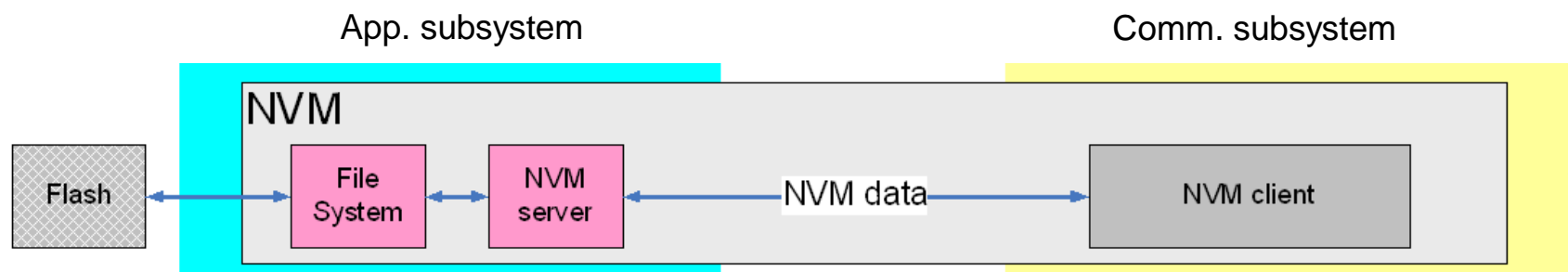


Marvell TZ / TEE

- ▶ TZ stands for Trust Zone, a section in DDR that considers secured.
- ▶ This is ARM feature allows configuring a range of capabilities to protect certain DDR sections in a way that will not allow Hackers to access it.
- ▶ **Not supported in this project**
- ▶ TEE – this is the SW package for TZ .

NVM

- ▶ NVM is data storage area in flash for CP usage.
- ▶ As CP cannot access directly to flash, AP is responsible to sync the actual code to flash (CP writes to DDR)
- ▶ Mainly used by the comm
 - Comm. configuration
 - RF calibration
 - Protocol stack related files
 - Audio configuration files
- ▶ **Possible design change**



Debug & Logging - DIAG

- ▶ Part of the Telephony. It is a diagnostic module that collect traces from the SW (MSA, APPS) and according to configuration:
 - Sends them over USB to a PC running the CATStudio tool
 - Saves them on the flash/SD card (offline logging).
 - Saves them in the DDR (cyclic buffers)
- ▶ Once the SD log files get to the PC (CATStudio, SD Card Reader, FTP) they can be opened using the CATStudio
- ▶ Currently we do not support diag over IP
- ▶ Solution for Linux HOST will be based on tty

Debug & Logging - Error Handler (EEH)

- ▶ COM EEH: handle COM errors and notify the APPS.
 - When the COMM has an unrecoverable event, the COMM DDR is written to the flash and can be read later via the debug/host I/F.
- ▶ APP EEH: handle APP errors, logging COM & APP error information in the memory and perform system recovery.
 - On an unrecoverable event, the entire DDR is written to the flash and can be read later via the debug/host I/F – (RAMDUMP).

Debug & Logging - Sulog

▶ Super Log

- ▶ Sulog is a new debug tool (HW) implemented in CP side and capable to debug both HW and SW events
- ▶ AP core is responsible to stream out Sulog log information from DDR using either:
 - ▶ USB
 - ▶ SD card
- ▶ AP Sulog application boots by default but doesn't effect system performance until actual Sulog debug begins

RUMDUMP

- ▶ RAMDump is a feature that upon a failure causes all the DDR area to be compressed and written to the memory.
- ▶ Contains:
 - One image (APPS, COMM & MSA) + a brief problem description file.
 - Complete information for APPS debug: COM debug info.
 - All Linux state is available: can analyze system state.
 - With Full kernel code/data: can analyze kernel state with symbol table.
 - All processes are available: can see call stack
 - Log available in kernel panic cases.
 - COM debug info.

RAMDUMP process

- ▶ SEM originated errors (COMM or APPS)
 - ▶ Stage 1 – preparation
 - Stream all debug info from COM side to APP side.
 - Compressed all logs in DDR.
 - Save APP debug info to DDR
 - Reset is preformed (**GPIO reset to keep DDR content**)
 - ▶ Stage 2 – copying
 - Uboot is running in special mode
 - Uboot compressed the DDR and copy it to SD card.
 - Full power reset is preformed
 - ▶ Stage 3 – Normal state
 - Normal boot
 - ▶ Example:
 - Apps assert: `echo p > /dev/ramdump_ctl OR eeh -T panic`
 - CP assert: `echo a > /dev/acipc OR eeh -T cpassert`



Debug & Logging

- ▶ SEM management
 - Config (stall/Ramdump)
 - Groups

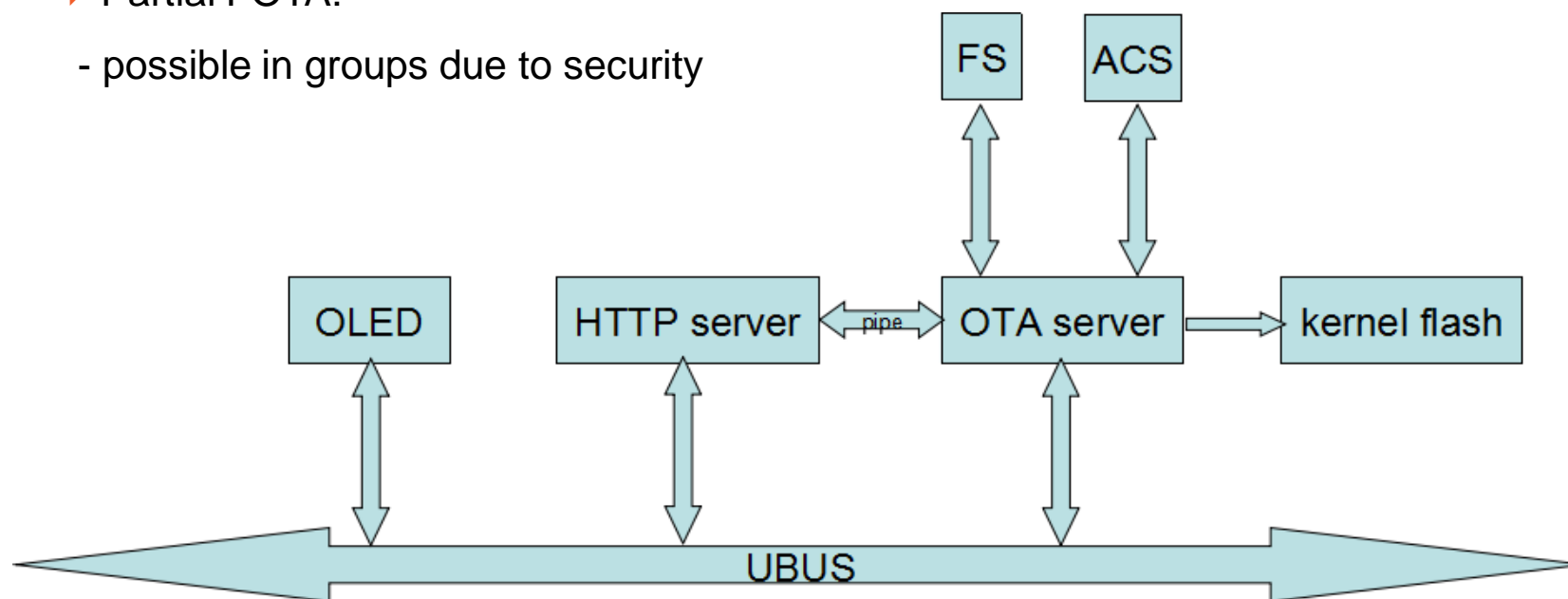
- ▶ GDB:
 - Gdb on target
 - Gdb on Linux machine

FOTA service

- ▶ Done in OBM
- ▶ OTAD is OTA-related daemon process. This process main responsibility is to do firmware upgrade involves common things. Such as FBF file integrity checking, Flash burning etc.
- ▶ OTAD interact with other APP through UBUS. OTAD implemented method call and notification.
- ▶ APP can launch firmware upgrade through method call. APP can listen notification get download process information.
- ▶ In addition to providing common service for other APP using, OTAD also achieve Marvell defined new firmware discovery protocol.

OTAD diagram

- ▶ End user has three ways to upgrade device:
 1. Device auto detect new firmware through internet. Such as TR069 ACS or other OTA protocol
 2. Upgrade device through WEB UI
 3. Firmware store on SD card
- ▶ flash flag is used for indication to OBM to perform FOTA process.
- ▶ Partial FOTA:
 - possible in groups due to security



SW Porting

- ▶ Pin Mux file for the new device
- ▶ Update kernel Device Tree (dts files)
- ▶ **Add a new profile** and Build accordingly using defconfig file
- ▶ Add specific HW support drivers etc. (optional)
- ▶ Add specific applications (optional)

Adding a new profile

1. Add a new profile entry in: “target/linux/mmp/pxa1826/profiles/marvell.mk”
2. Create a dts file for the board, dts_filename.dts in: “/marvell/linux/arch/arm/boot/dts”
3. Add “dts_filename.dtb” to create dtb file in: “arch/arm/boot/dts/Makefile”
4. Create a defconfig file in “config/” with your desired configuration,
5. Add new entry in “target/linux/mmp/image/Makefile”:
6. Align “rules.mk” with the new profile

► Additional steps (optional):

If you want to be able to identify your board from the console (TeraTerm),

Then, in file: “target/linux/mmp/base-files/lib/pxa182x.sh”

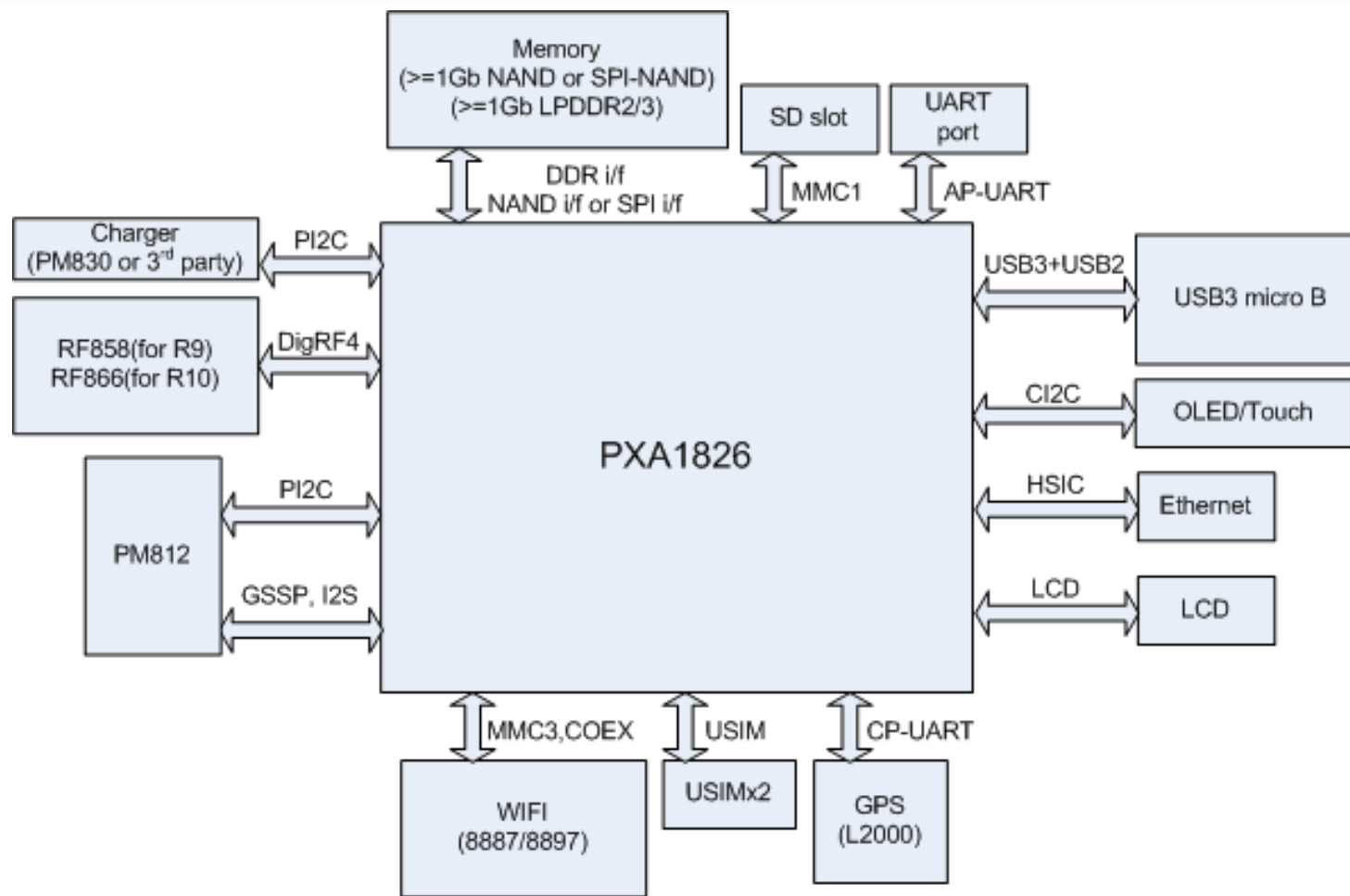
Function: pxa182x_board_detect()

You need to add a case for your board (profile).



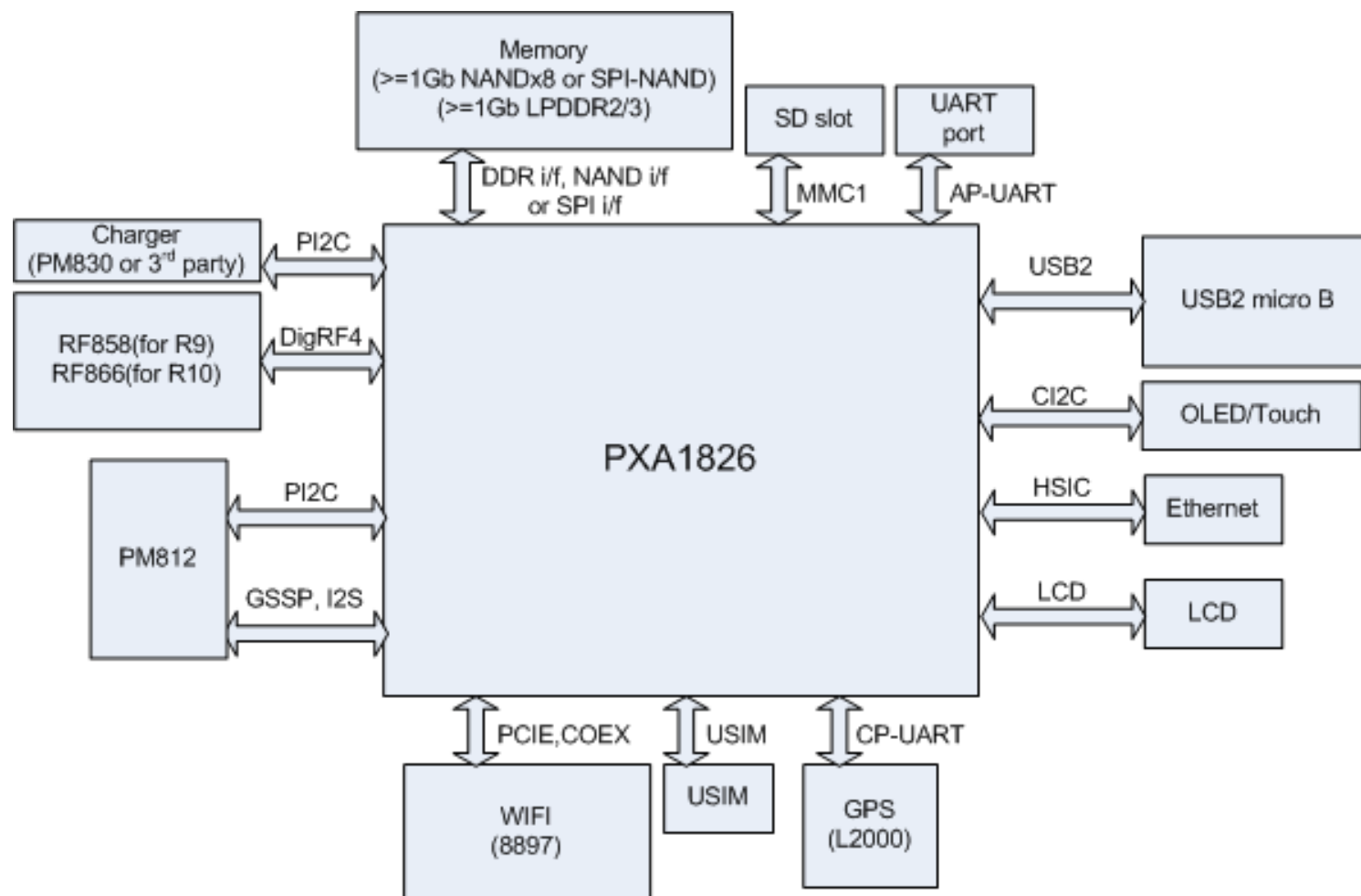
Back-up

Hardware Block Diagrams - Mainstream configuration #1



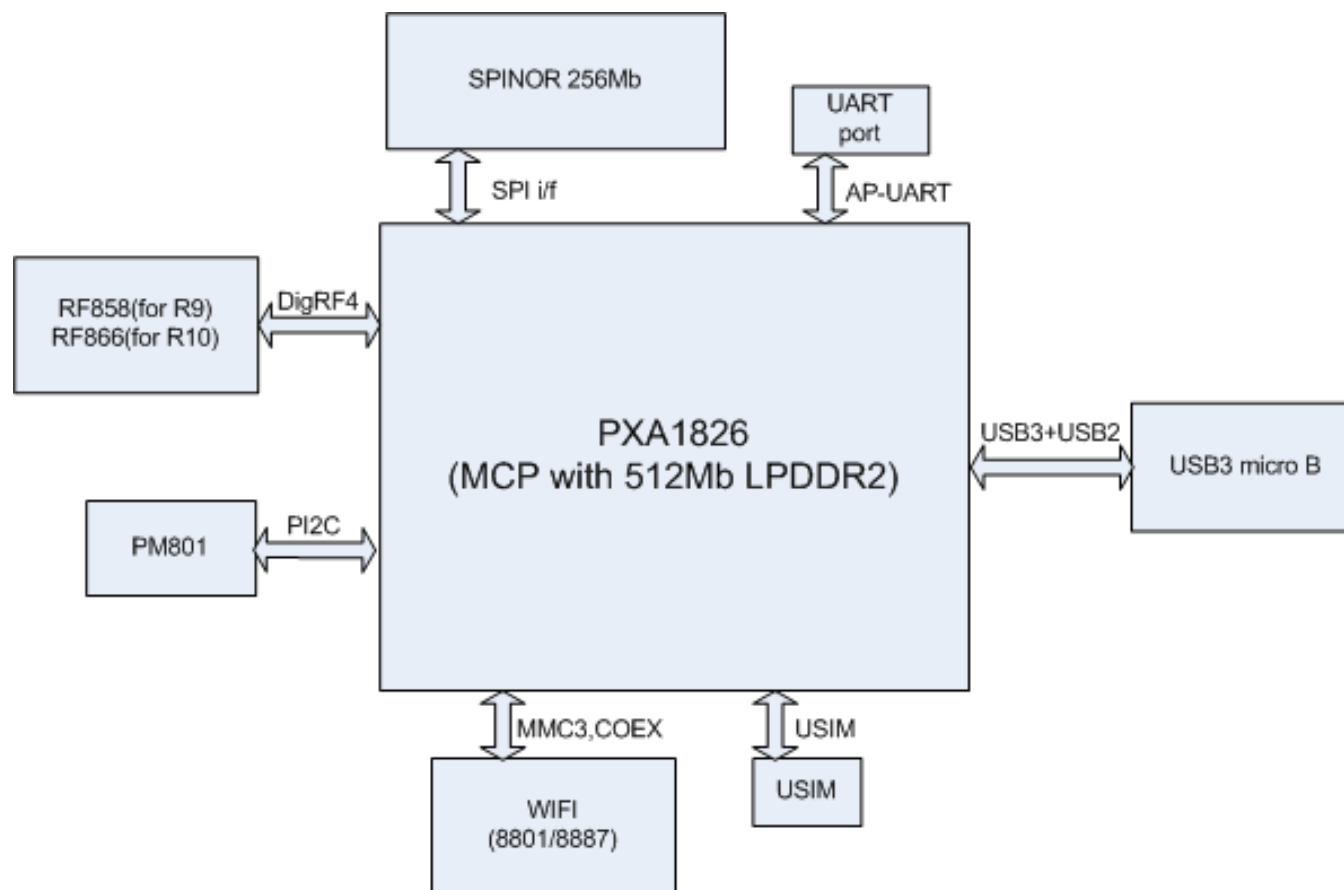
- ▶ WIFI is using SDIO, <300Mbps
- ▶ Support USB3

Hardware Block Diagrams – Mainstream configuration #2



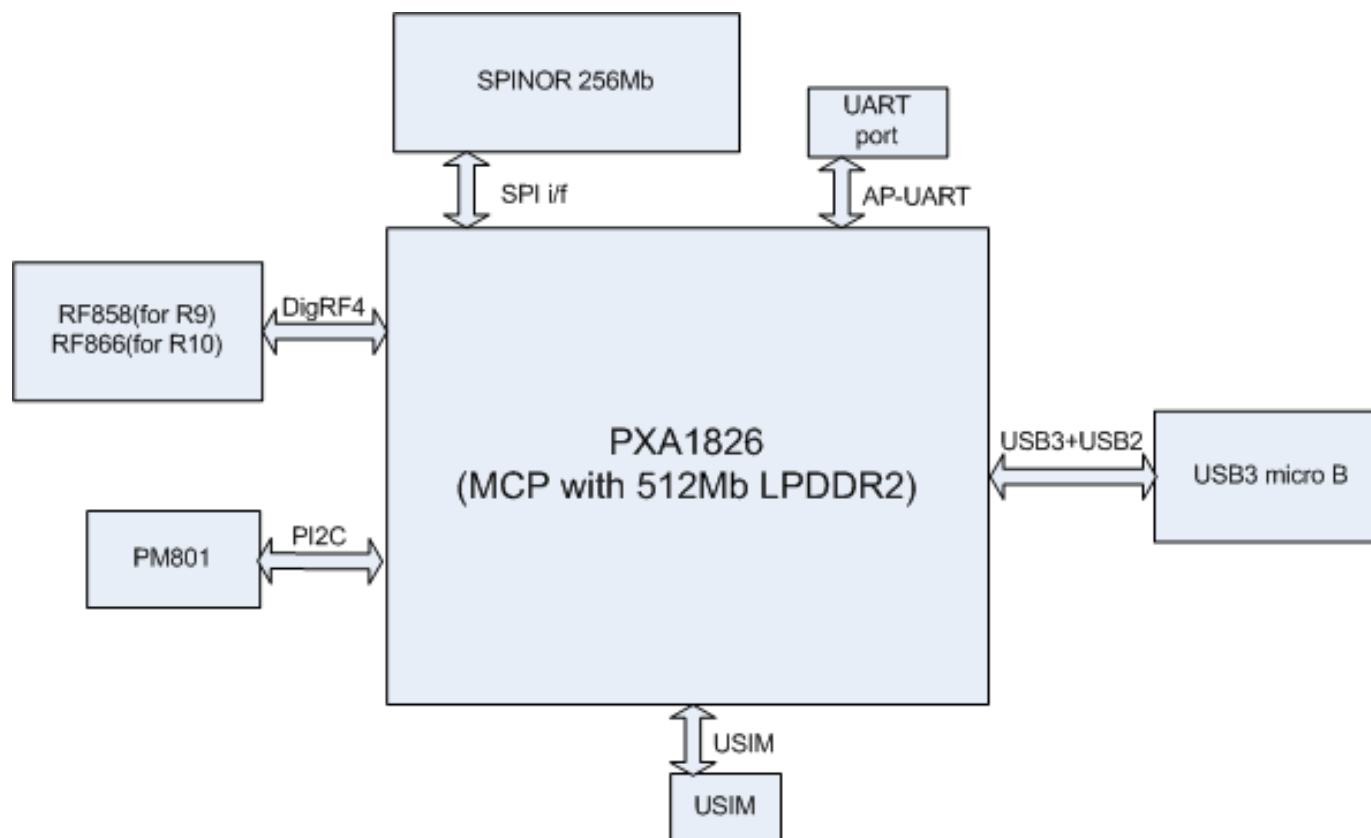
- ▶ **WIFI is using PCIe with 8897, up to 500Mbps**
- ▶ **No USB3**

Hardware Block Diagrams – Lowest MIFI configuration #3

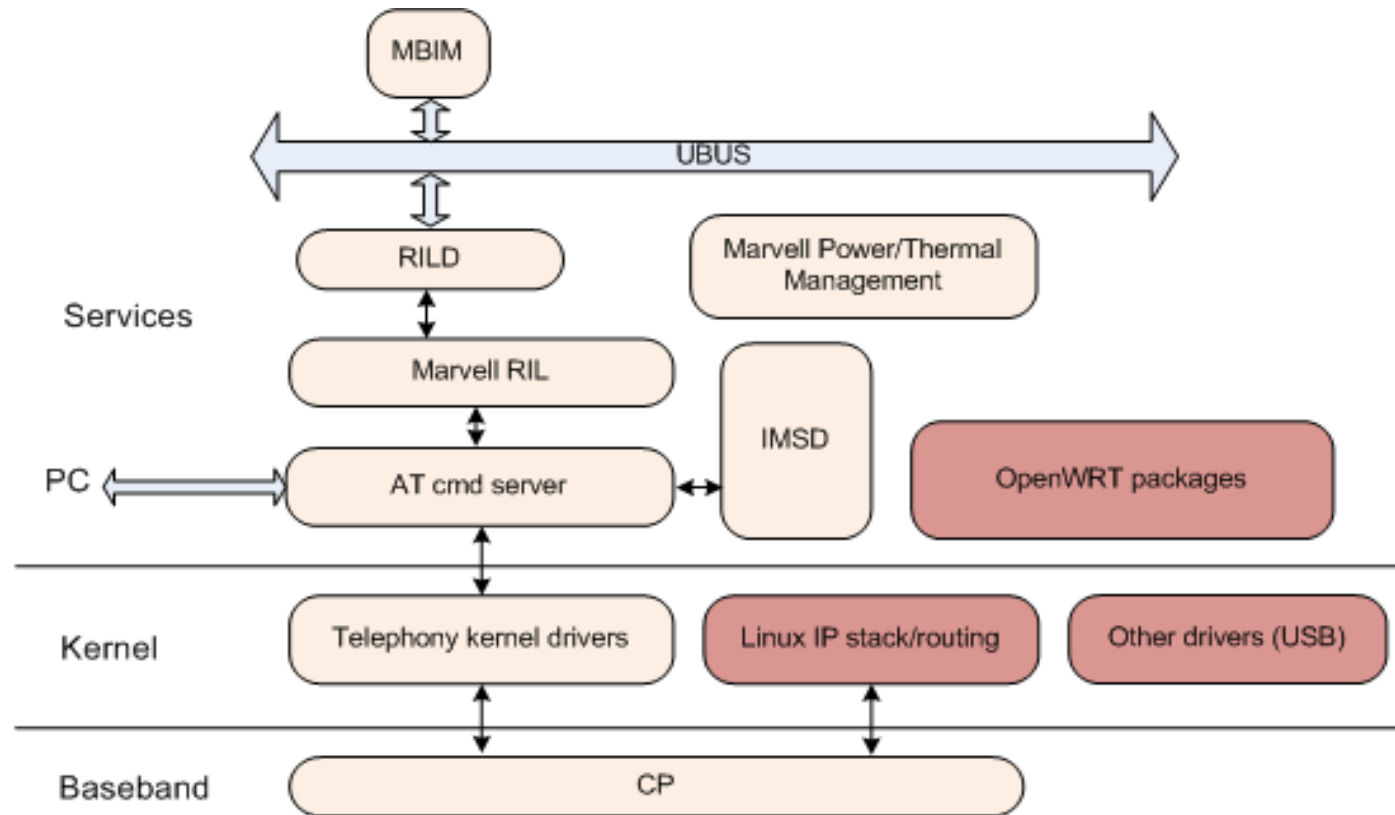


- ▶ SW Feature are limited
- ▶ Only have 24MB DDR

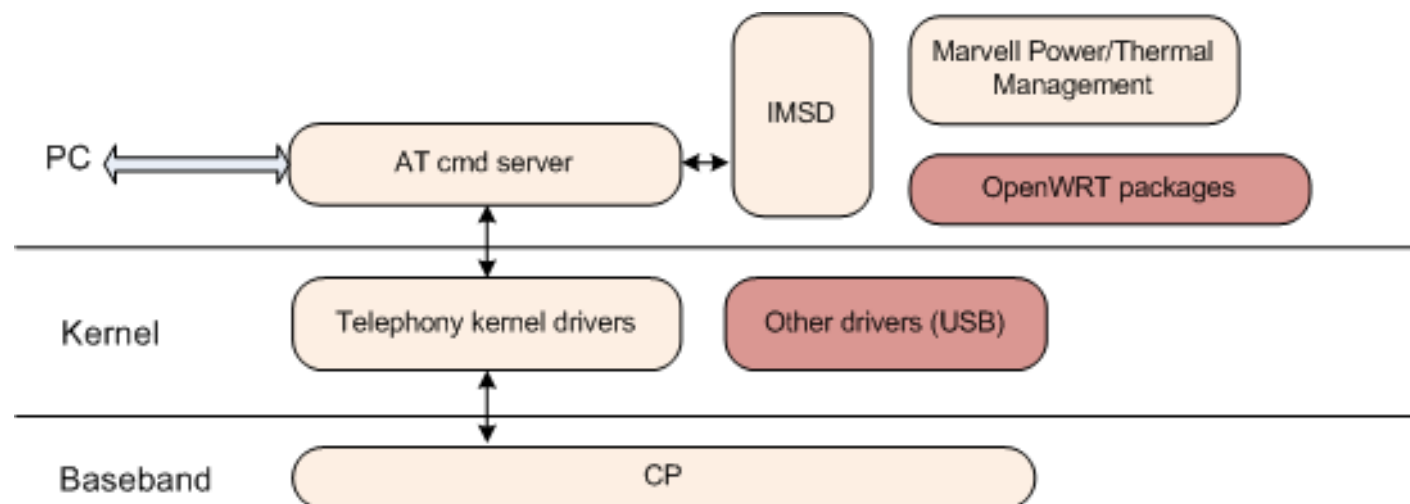
Hardware Block Diagrams – Dongle/Module configuration #4



Software Block Diagram – Dongle with MBIM

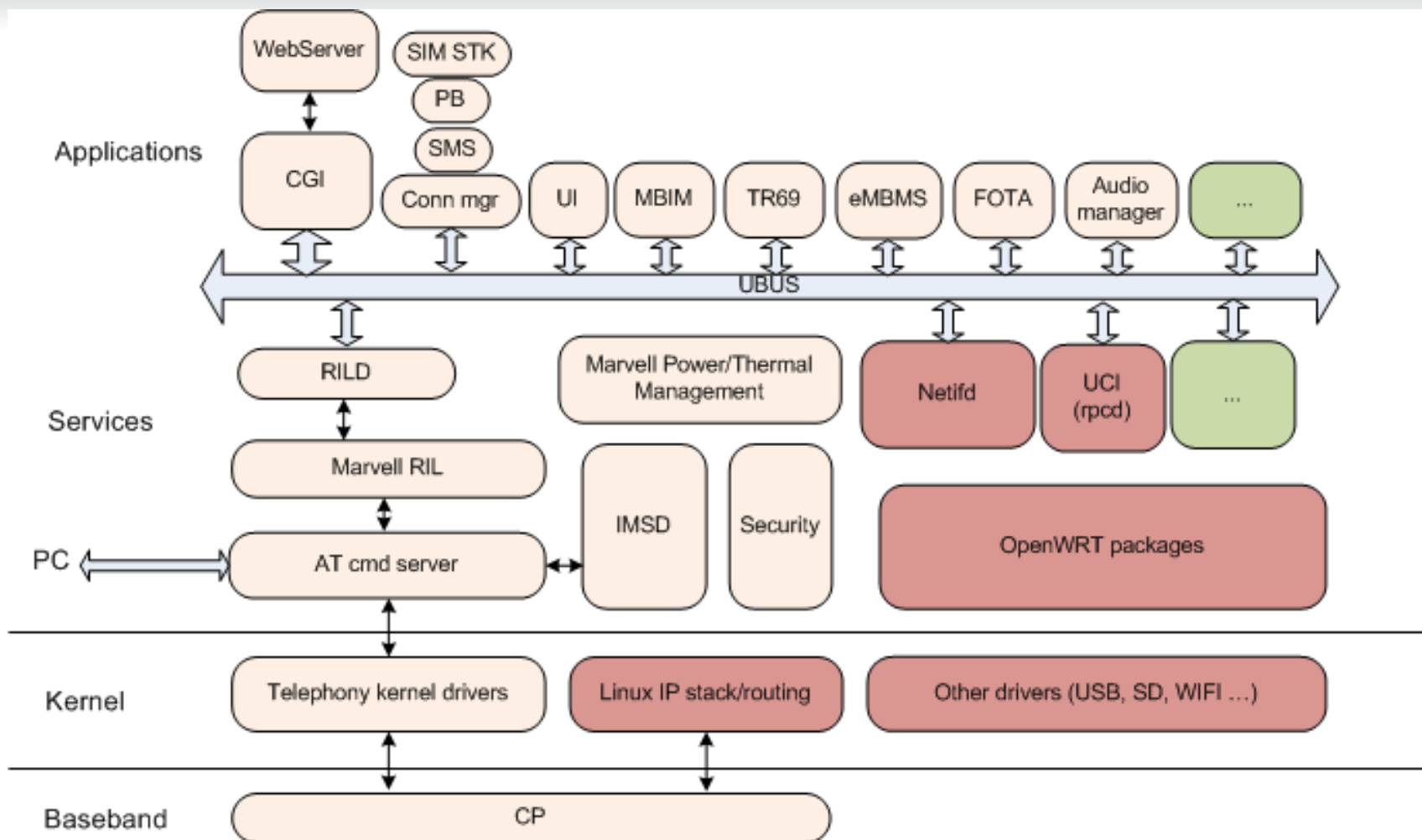


Software Block Diagram – Dongle without MBIM



- ▶ This is for customers who only need modem and the TCP/IP/NAT is not in 1826 side

Software Block Diagram – Mainstream and Lowest MIFI



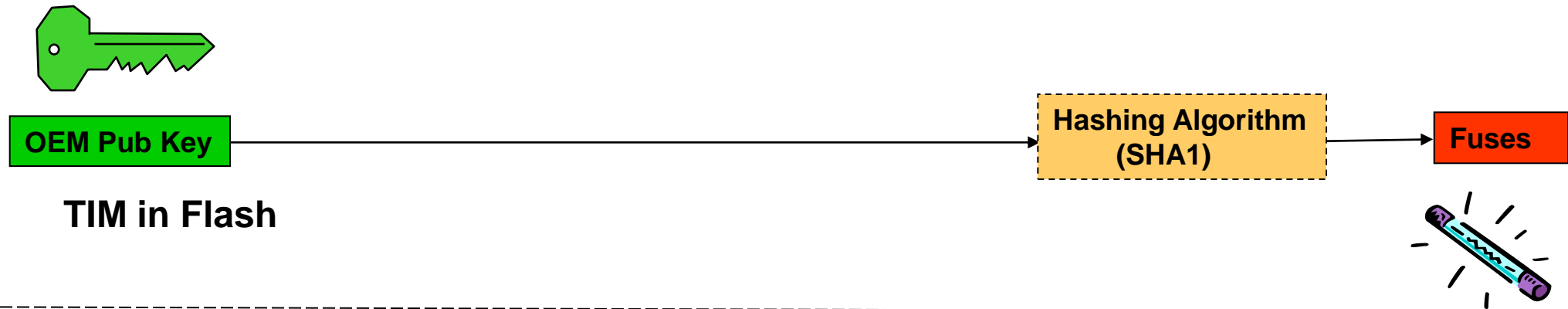
► For lowest MIFI, architecture is same but the features will be limited

MRD API

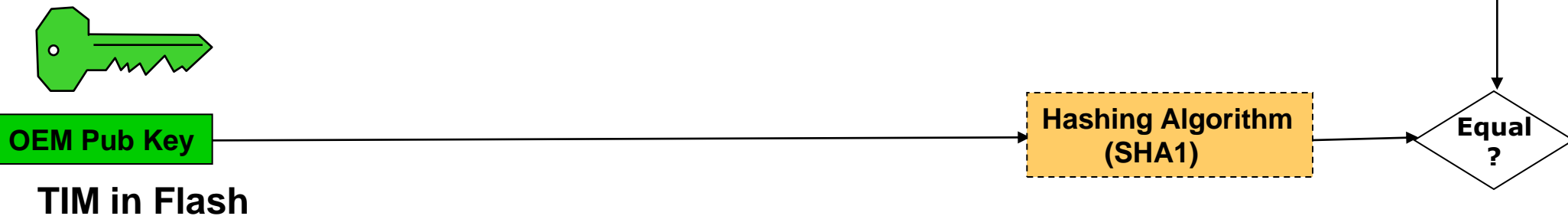
- ▶ **The MRD API is in charge of the following tasks:**
 - General purpose: Read “file” from MRD.
 - General purpose: Add “file” to MRD.
 - General purpose: Remove “file” from MRD.
- ▶ **All the commands are received from the AT server.**
- ▶ **The MRD API is a service of the telephony process which is executed as follows:**
 - r/w in production mode.
 - r/o in normal mode.

Platform Binding and Verification (on Target)-

Production mode -ONLY



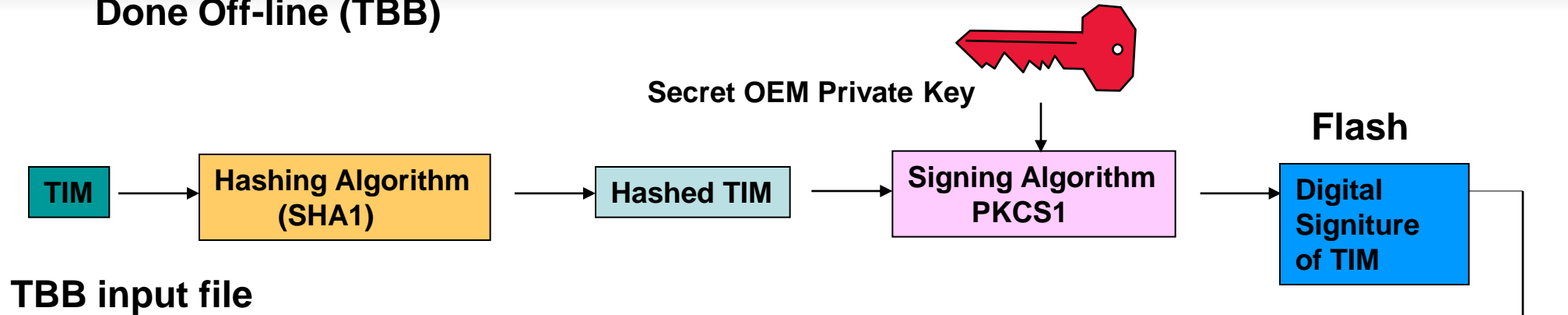
Each boot



TIM Authentication

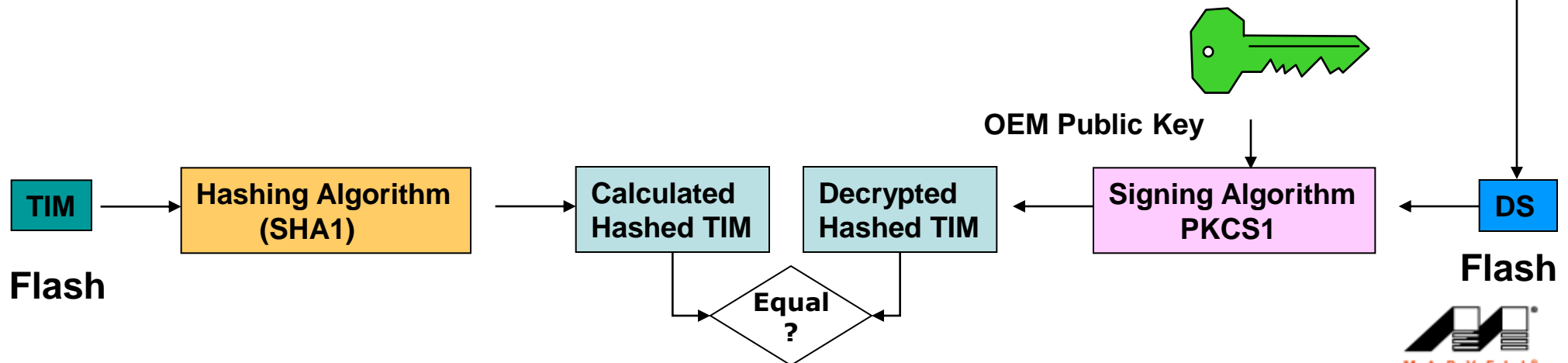
Production

Done Off-line (TBB)



On Target

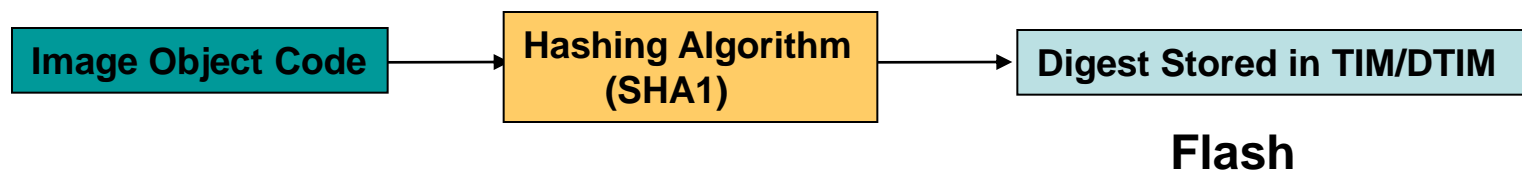
Each boot



All other Image Verification

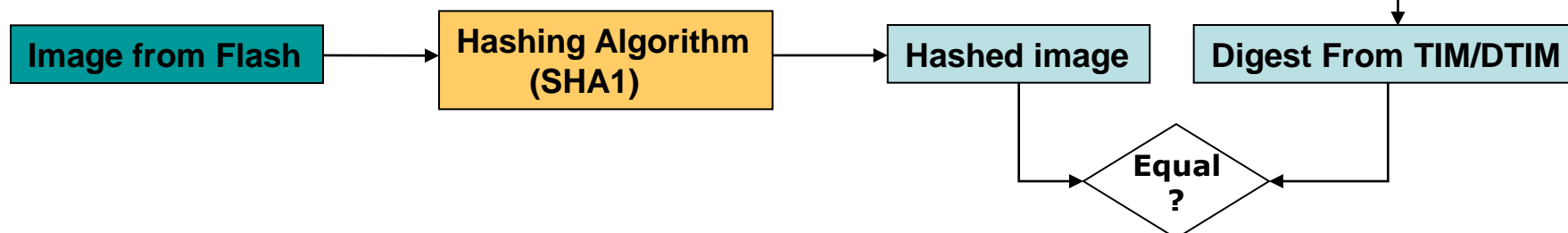
Done Off-line (TBB)

Production



On Target

On each boot, for every image in TIM/DTIM



Netifd

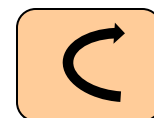
netifd is an [RPC](#)-capable [daemon](#) written in [C](#) for better access to kernel APIs with the ability to listen on [netlink events](#). Netifd has replaced the old *OpenWrt-network configuration scripts*, the actual scripts that configured the network, e.g.

`/lib/network/*.sh,`

`/sbin/ifup`

some scripts in `/etc/hotplug.d.`)

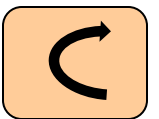
netifd is intended to stay compatible with the existing format of [/etc/config/network](#), the only exceptions being rare special cases like aliases or the overlay variables in `/var/state` (though even most of those can be easily emulated).



The UCI System

The abbreviation UCI stands for *Unified Configuration Interface* and is intended to centralize the configuration of OpenWrt.

Configuration should be easy and straightforward, making life easier! UCI is all about that.



ipk file

IPK, or Itsy Package is a compressed archive format file derived from the Debian package format. It is used for handheld software installations.

Life cycle states

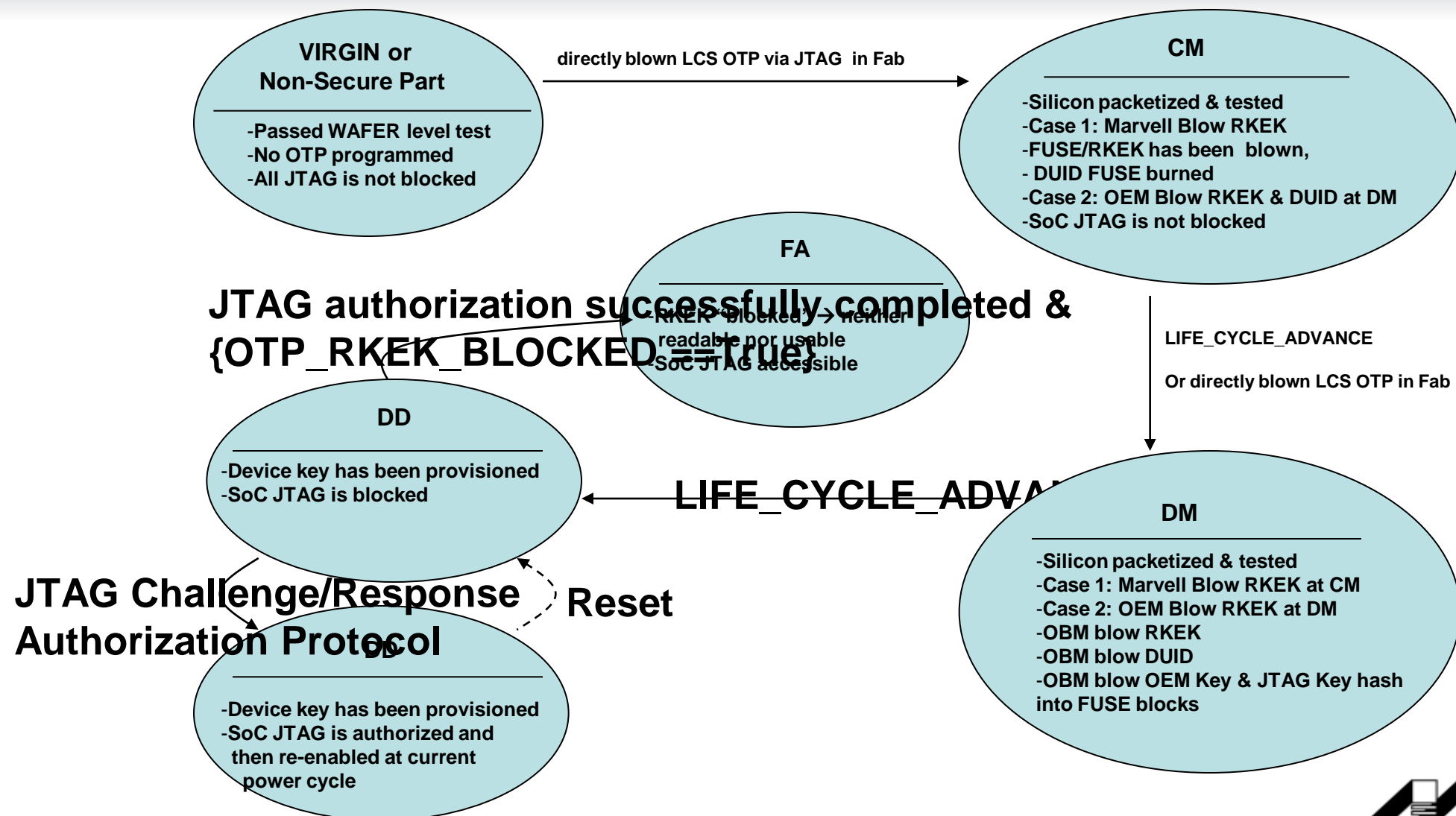
- Life cycle states (LCS):

1. Chip is manufactured - LCS = 0, does not contain any secret (key). JTAG debug is enabled.
2. After initial provisioning (Marvell FAB) - LCS = CM (1), RKEK exist in the chip
3. After customer provisioning – LCS = DM (2), public key is burned to fuses & boot memory is selected.
4. Product is handed for end user – LCS =DD (3), JTAG is disabled.
5. OEM sends the deceive to Marvell as RMA – LCS =FA (4). JTAG is enabled.

RKEK use, RKEK read, and JTAG availability Vs LCS

	New Chip (All Zero)	CM Chip Manufacturing	DM Device Manufacturing	DD Device Deployment	FA Failure Analysis	JPD JTAG Permanently Disabled
RKEK Use	Zero Value	Programmed Keys	Programmed Keys	Programmed Keys	Not loadable	Programmed Keys
RKEK Read	Readable	Readable / Not Readable Depends on ap[61]	Readable / Not Readable Depends on ap[61]	Not Readable with ap[61] blown	Not Readable	Not Readable
Seagull JTAG	Available	Available	1) Available 2) Not Available at power up but SW may enable per power cycle	1) Available 2) Not Available at power up but SW may enable per power cycle	Available	Not Available
Cortex JTAG	Available	Available	1) Available 2) Not Available at power up but SW may enable per power cycle	1) Available 2) Not Available at power up but SW may enable per power cycle	Available	Not Available

LCS State Transition & JTAG Access Status



- ▶ **OMA-DM** - Open Mobile Alliance (OMA) Device Management (DM). Mmanagement of mobile devices. Device management is intended to support the following uses:
 - Provisioning – Configuration of the device (including first time use), enabling and disabling features
 - Device Configuration – Allow changes to settings and parameters of the device
 - SW Upgrades – Provide for new software and/or bug fixes to be loaded on the device, including applications and system software
 - Fault Management - Report errors from the device, query about status of device