MEDIATEK

Android Driver Allinone





Outline

Overview

- ***Chip***
- Build Command

Boot Up

- Bootloader

Lights System

Battery Manager

- Battery Service
- Battery Charging
- Power off charging

Input/Output

- Keypad
- Touch
- Display

Connecting

- WIFI
- MT6620 Customization

Sensor System

- Sensor Hal
- Sensor Driver Customization

Audio

Misc.

- GPIO
- I2C
- PMIC
- Headset

Appendix

- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

MT6516 Overview (1/2)

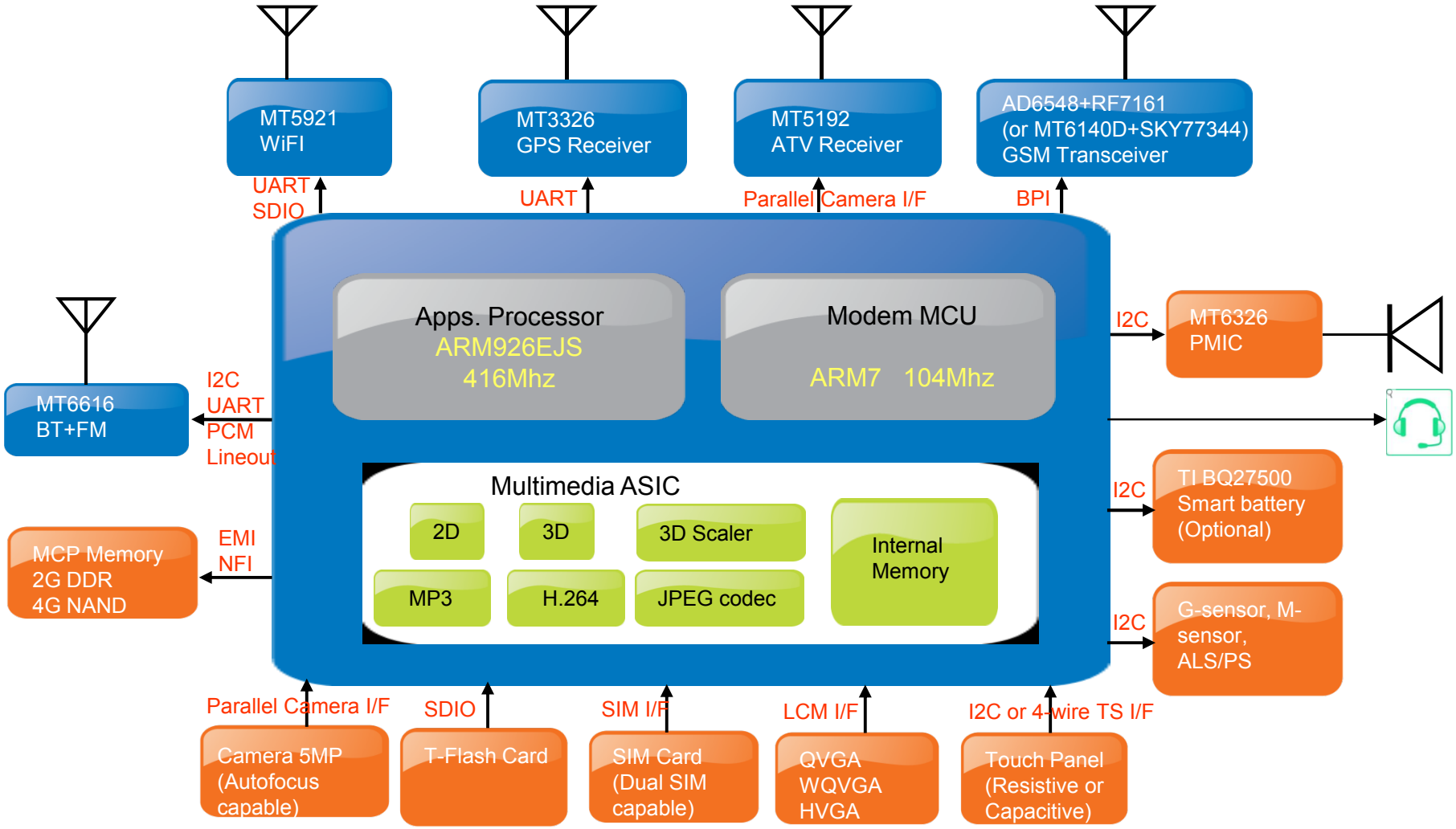
- **Application CPU**
  - ARM926EJS 416MHz
  - 32KB I-Cache, 32KB D-Cache
  - 16KB I-TCM, 16KB D-TCM
  - MMU support
- **Modem CPU**
  - ARM7EJS 104MHz
  - GSM/GPRS/EDGE Modem Class 12
- **Multi-Media DSP**
  - Video DSP 312MHz with 128KB L1 cache and 64KB L2 cache
- **Bus System**
  - 1 high performance external memory bus support 32-bit DDR memory at 104MHz
  - 2 64-bit high performance internal bus for multimedia
- **Built-in ROM/RAM**
  - Boot ROM for booting
  - 304KB SRAM
- **Operating Conditions**
  - Core voltage: 1.3V
  - I/O voltage: 2.8V
  - Memory: 1.8V/2.8V
  - NAND: 1.8V/2.8V
  - LCM interface: 1.8V/2.8V
  - Clock source: 13MHz, 32.768KHz
- **Peripheral**
  - 31 DMA channels
  - Watch dog timer
  - 7 advanced general purpose timer
  - 7 PWMs
  - 9 AuxADC channels
  - 8x8 Qwerty kaypad
  - Touch panel

MT6516 Overview (2/2)

- **Connectivity**
  - USB 2.0 8 Tx and 8 Rx endpoints
  - 4 high-speed UARTs
  - 3 I2C channels
  - 3 SDIO/SD/SDHC/MS/MSPRO/MMC interface
  - Dual SIM card interface
  - I2S interface
- **Display**
  - Support host interface, RGB interface, and MIPI DSI high speed serial interface
  - Resolution up to WVGA*1
- **Graphic**
  - 3D Graphics, OpenGL ES 1.1
  - 2D Graphics HW acceleration
  - Hardware PNG decoder
- **Image**
  - Built-in image signal processing for 5M pixel camera sensor
  - Hardware JPEG encode/decode
- **Audio**
  - Programmable low-pass IIR filter to eliminate noise
  - Sample rate conversion up to 48KHz
  - I2S interface
  - Audio encode: AMR-NB
  - Audio decode: MP3, AMR, WB-AMR+, WMA, WMA+, BSAC, AAC, AAC+
- **Video**
  - H.264 video decoder D1 30fps
  - MPEG4/H.263 video encoder D1 30 fps
  - MPEG4/H.263 video decoder D1 30fps
  - VC-1 video decoder CIF 30fps (by DSP)
  - RealVideo decoder CIF 30fps (by DSP)

*1 MT6516 Android only supports up to HVGA resolution

MT6516 Design Building Blocks



MTK HSPA Smartphone Feature Overview

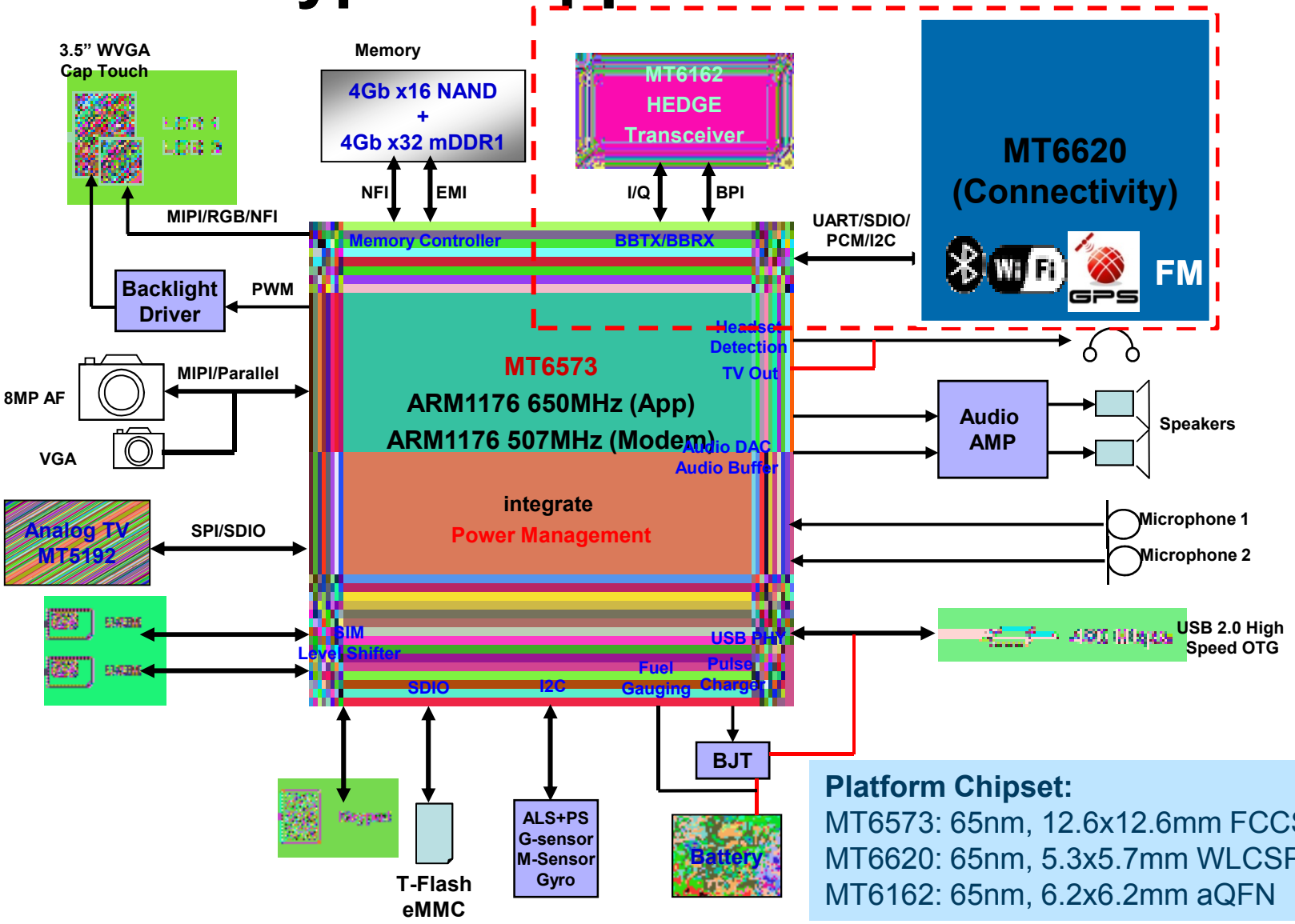
HSPA Smartphone Platform:

MT6573 + MT6162 + MT6620

MT6573 Features:

- App. Processor: ARM1176JZF-S **650 MHz**
  - Dedicated 32KB I-Cache and 32KB D-Cache
  - DVFS 1.0 ~ 1.35V, step 25mV support
  - ARM™ TrustZone® Security
- Modem Processor: ARM1176JZ-S **507 MHz**
  - HSDPA, Downlink 7.2Mbps (Category 8)
  - HSUPA, Uplink 5.76Mbps (Category 6)
  - UMTS Band 800/850/900/1700/1900/2100/AWS
  - EDGE Qual-Band 850/900/1800/1900
- PMU/R-touch panel driver integrated
- Supports up to FWVGA (864x480) LCM
  - 24bit CPU/RGB IF, 2-lane MIPI DSI I/F
- MFlexVideo driven multimedia engine
  - MPEG-1/2/4, DivX, H.264, VC1/WMV9 **FWVGA 30fps decode**
  - MP4/H.263 FWVGA 30fps encode.
  - QCIF Video Telephony
- Camera ISP: 8MP
  - MIPI/Parallel I/F, EIS, face detection
- Advanced DSP acoustic functionality
  - AEC and noise reduction
- Supports QWERTY keypad
- Multi-memory system with more flexibility for phone design, which supports
  - NAND-boot , eMMC boot
  - NAND data storage
  - LPDDR SDRAM 200MHz
- Integrate DC couple headphone driver
- Integrate dual microphone / digital microphone I/F
- Integrated fuel gauge.
- Supports SD, MMC, Memory Stick, Memory Stick Pro, SDIO cards
- Supports dual SIM
- Built-in USB 2.0 (High-speed/OTG), IrDA
- Package size: 12.6x12.6 mm^2 , 518 balls TFBGA with 0.4 mm pitch

MT6573 Typical Application



Platform Chipset:
MT6573: 65nm, 12.6x12.6mm FCCSP
MT6620: 65nm, 5.3x5.7mm WLCSP
MT6162: 65nm, 6.2x6.2mm aQFN

MT6516 / MT6573 Feature Comparison

	Mediatek MT6516	Mediatek MT6573
Package	15x15mm 564-pin, BGA, 0.5mm, 65nm	12.6x12.6mm 518 balls, BGA, 0.4mm, 65nm
Apps Processor	ARM926EJS@ 416 MHz w/ 32KB/32KB I/D cache	ARM1176JZFS@ 650 MHz w/ 32KB/32KB I/D cache 128KB L2, DVFS support
Modem Processor	ARM7EJ-S @ 104 MHz , 104MHz DSP	ARM1176JFS@ 507 MHz, 280MHz DSP
Modem	GPRS, EDGE class12	EDGE class12, HSDPA Cat8 7.2Mbps, HSUPA Cat6 5.76Mbps
Memory	104MHz-mDDR, 128MB x 4, NAND (NFI)	200MHz-mDDR, 256MB x 4, NAND (NFI)
Camera / TV-out	5MP Bayer/YUV, 10bit parallel, MIPI CSI-2	8MP Bayer/YUV, 10bit parallel, MIPI CSI-2 CVBS TV out
Display	WVGA, 24-bit color, MIPI DSI, CPU/RGB I/F	FWVGA, 24-bit color, MIPI DSI, NFI, CPU/RGB I/F
Audio	64-Poly, MP3,AAC,HE-AAC, WMA,G.711,G.723.1,G.729,AWB+,3D effect	64-Poly, MP3,AAC,HE-AAC, WMA,G.711,G.723.1,G.729,AWB+,3D effect Dual mic, Digital mic support
Video Decode	MPEG4/H.264: D1 @ 30fps	MPEG4/H.264: FWVGA @ 30fps
Video Encode	MPEG4: D1 30 fps	MPEG4: FWVGA @ 30fps, H.264 CIF @ 30fps
Video Telephony	NA	3G-324M: QCIF 15 FPS, 64kbps
Video Streaming	QVGA Video Streaming	MPEG4/H.264 to D1, 30 fps
Peripherals	UARTx4, SIMx2, R-touch, PCMx1, I2Sx2, I2Cx3, SPIx1 SDIOx3, Key Matrix 8x8 USB 2.0 HS OTG int. PHY x 1	UARTx4, SIMx2, R-touch, PCMx1, I2Sx2, I2Cx2, SPIx1 SDIOx4 , Key Matrix 8x8 USB 2.0 HS OTG int. PHY x 1, USB FS host x 1
Power Management	External PMIC MT6326	Integrate PMU, 5 bucks, 20 LDOs, LED driver, pulse charger, gas gauge

Outline

Overview

- Chip
- ***Build Command***

Boot Up

- Bootloader

Lights System

Battery Manager

- Battery Service
- Battery Charging
- Power off charging

Input/Output

- Keypad
- Touch
- Display

Connecting

- WIFI
- MT6620 Customization

Sensor System

- Sensor Hal
- Sensor Driver Customization

Audio

Misc.

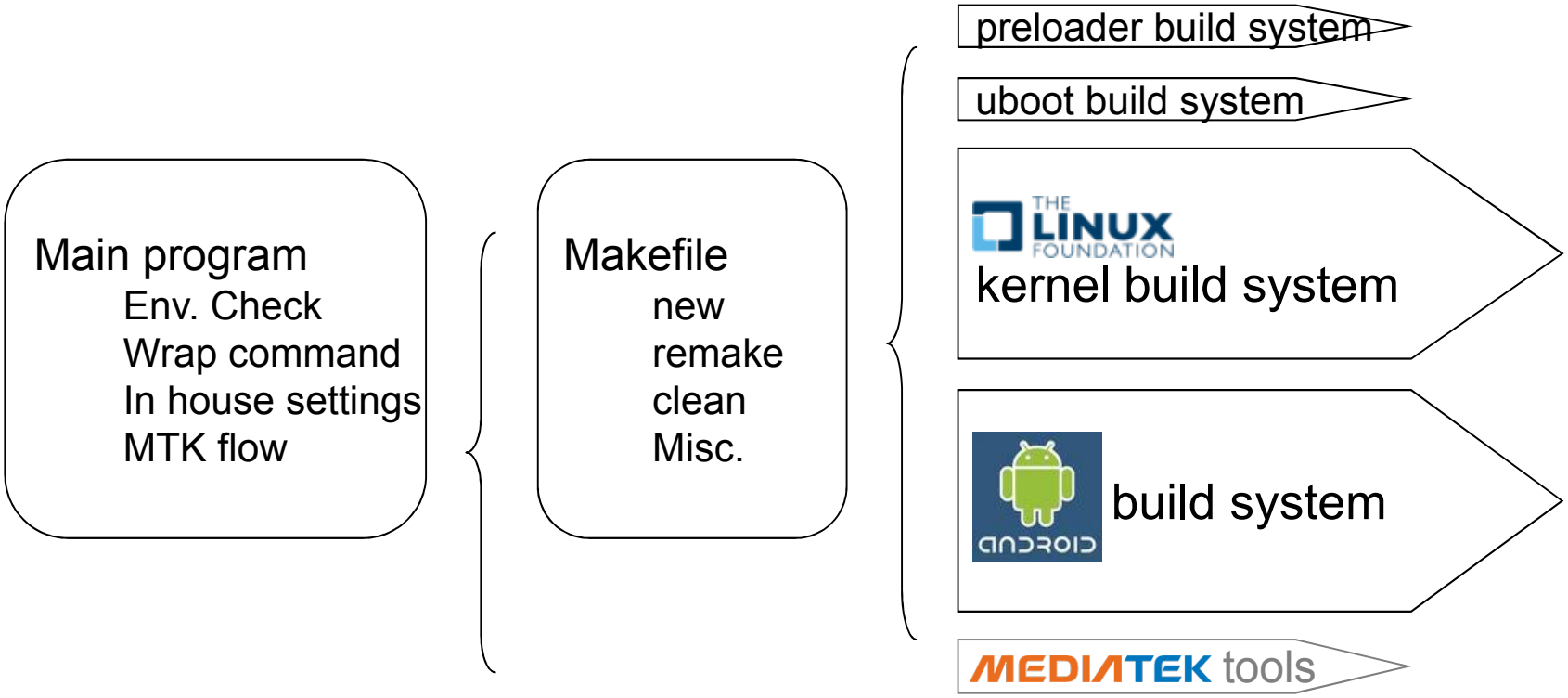
- GPIO
- I2C
- PMIC
- Headset

Appendix

- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

A Start Build (1/4)

- main entry program: makeMtk



A Start Build (2/4)

- Support straightforward function command to wrap build commands arguments

makeMtk [PROJECT] ACTION [MODULES]

- abbrev mk, **show help** if without any arguments
- PROJECT
  - gw616, ds269, oppo, mt6516_evb, emulator
- ACTION
  - new, clean, remake, parse, bm_new, bm_remake, listproject, codegen, gen_cust
    - abbrev: n, c, r, p
- MODULES
  - bootloader, kernel, android, preloader
    - abbrev: bl, k, dr, pl

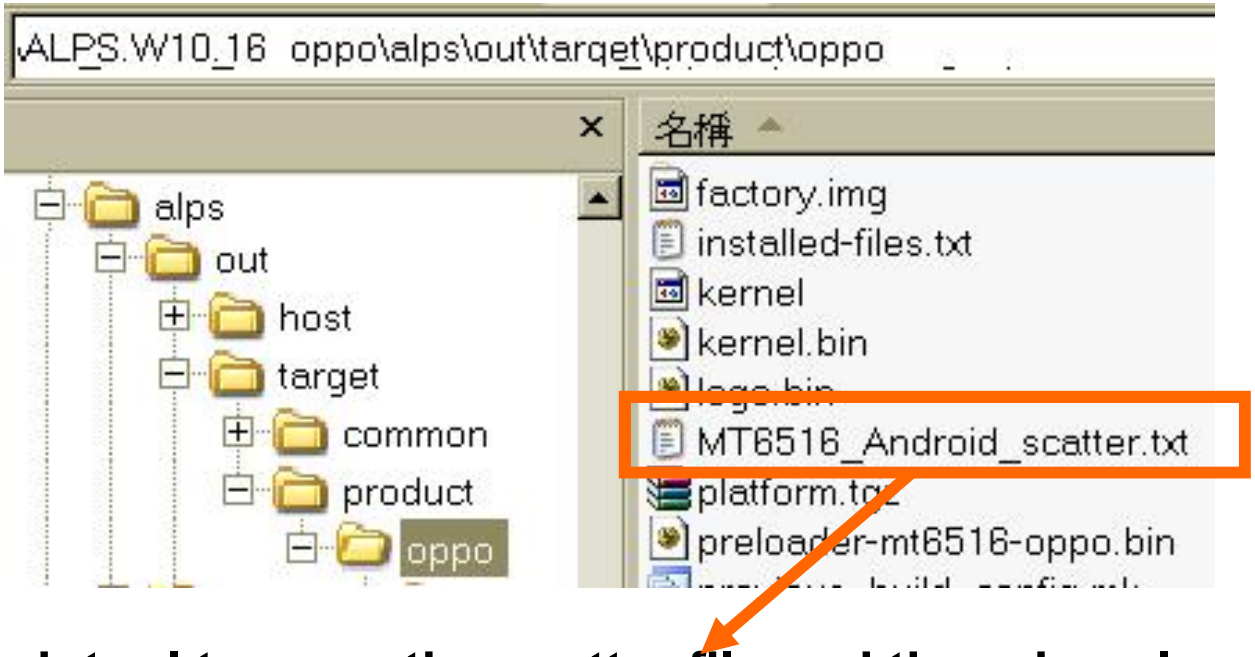
A Start Build (3/4)

- Use makeMtk for the codebase: **makeMtk oppo bm_new**

```
2010/04/06 08:04:47 cleaning preloader...
                    LOG: out/target/product/oppo_preloader.log
                    ==> [OK]    2010/04/06 08:04:47
2010/04/06 08:04:47 cleaning uboot...
                    LOG: out/target/product/oppo_uboot.log
                    ==> [OK]    2010/04/06 08:04:57
2010/04/06 08:04:57 cleaning kernel...
                    LOG: out/target/product/oppo_kernel.log
                    ==> [OK]    2010/04/06 08:05:33
2010/04/06 08:05:33 cleaning android...
                    LOG: out/target/product/oppo_android.log
                    ==> [OK]    2010/04/06 08:05:36
2010/04/06 08:05:36 gen_custing ...
                    LOG: out/target/product/oppo_gen_cust.log
                    ==> [OK]    2010/04/06 08:05:37
2010/04/06 08:05:37 codegening ...
                    LOG: out/target/product/oppo_codegen.log
                    ==> [OK]    2010/04/06 08:05:44
2010/04/06 08:05:44 update-apiing android...
                    LOG: out/target/product/oppo_update-api.log
                    ==> [OK]    2010/04/06 08:12:54
2010/04/06 08:12:54 building preloader...
                    LOG: out/target/product/oppo_preloader.log
                    ==> [OK]    2010/04/06 08:13:00
2010/04/06 08:13:00 building uboot...
                    LOG: out/target/product/oppo_uboot.log
                    ==> [OK]    2010/04/06 08:14:29
2010/04/06 08:14:29 building kernel...
                    LOG: out/target/product/oppo_kernel.log
                    ==> [OK]    2010/04/06 08:23:59
2010/04/06 08:23:59 building android...
                    LOG: out/target/product/oppo_android.log
                    ==> [OK]    2010/04/06 09:46:35
```

A Start Build (4/4)

- bm_new is a clean build of Yusu project
  - Take around 1h40m
- After a successful build, many files will be generated @ alps/out/target/product/oppo



- Use flash tool to open the scatter file and then download

A Quick Build

- Use Android native build command
  - 1.a source build/envsetup.sh @ <alps>
  - 1.b TARGET_PRODUCT=<project> m
    - Take long time > 10 min

Or

  - 2.a source build/envsetup.sh @ <alps>
  - 2.b TARGET_PRODUCT=<project> mmm <my_module_path>
    - 3 min
    - Same as
      - cd <my_module_path>
      - TARGET_PRODUCT=<project> mm
  - 2.c TARGET_PRODUCT=<project> m snod
    - Will pack system.img
    - 3 min

Some Useful Command

- Build Bootimage
  - 1. makeMtk <project_name> remake kernel
  - 2. TARGET_PRODUCT= <project_name> mk bootimage
- Generate DCT code
  - makeMtk mt6516_evb codegen
- Synchronize MTK folder's sourcecode
  - makeMtk mt6516_evb gen_cust
- Build release version binary
  - makeMtk -opt=TARGET_BUILD_VARIANT=user <project_name> new

Outline

OverView
- Chip
- Build Command

Boot up
- ***Bootloader***

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power off charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

Audio

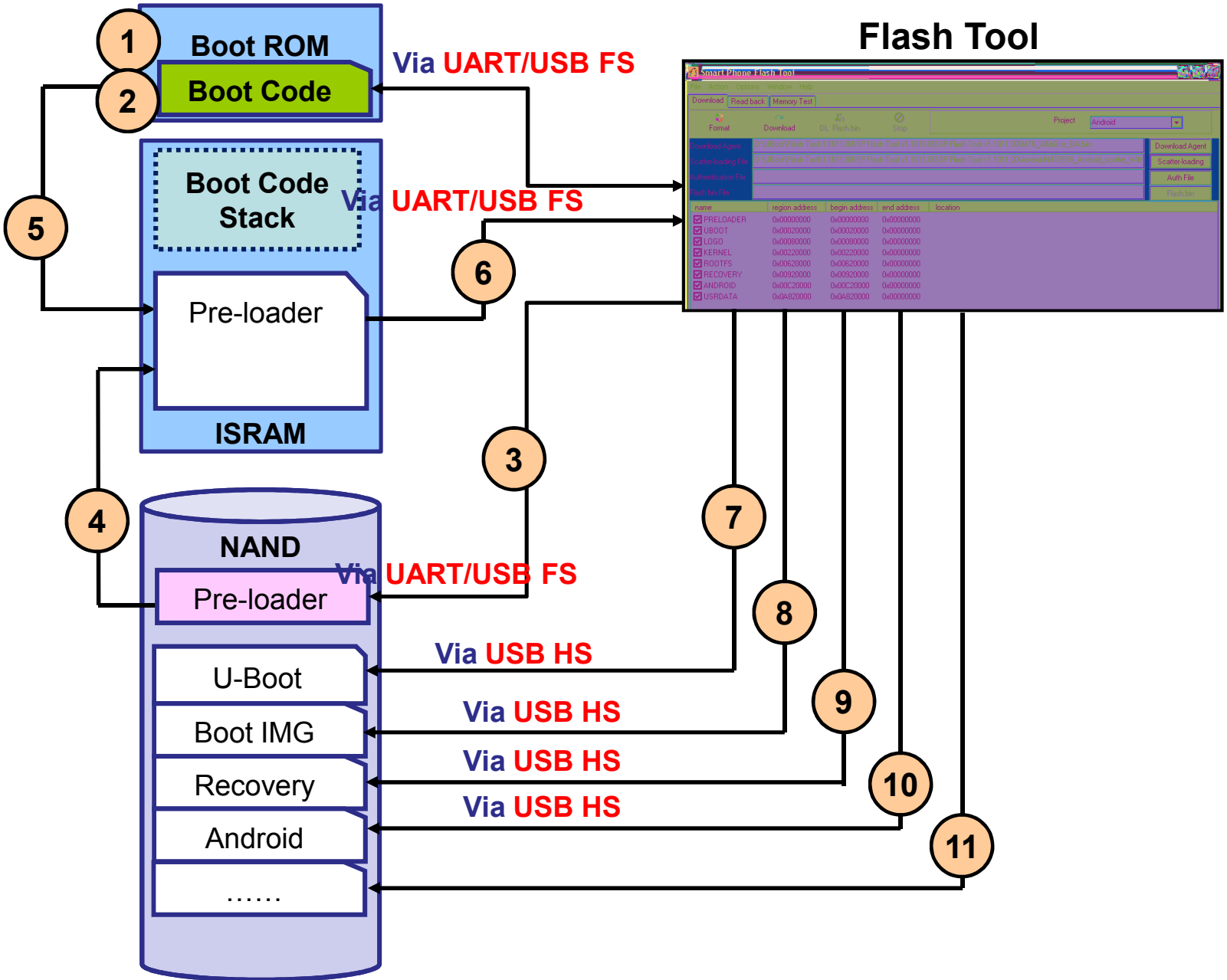
Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

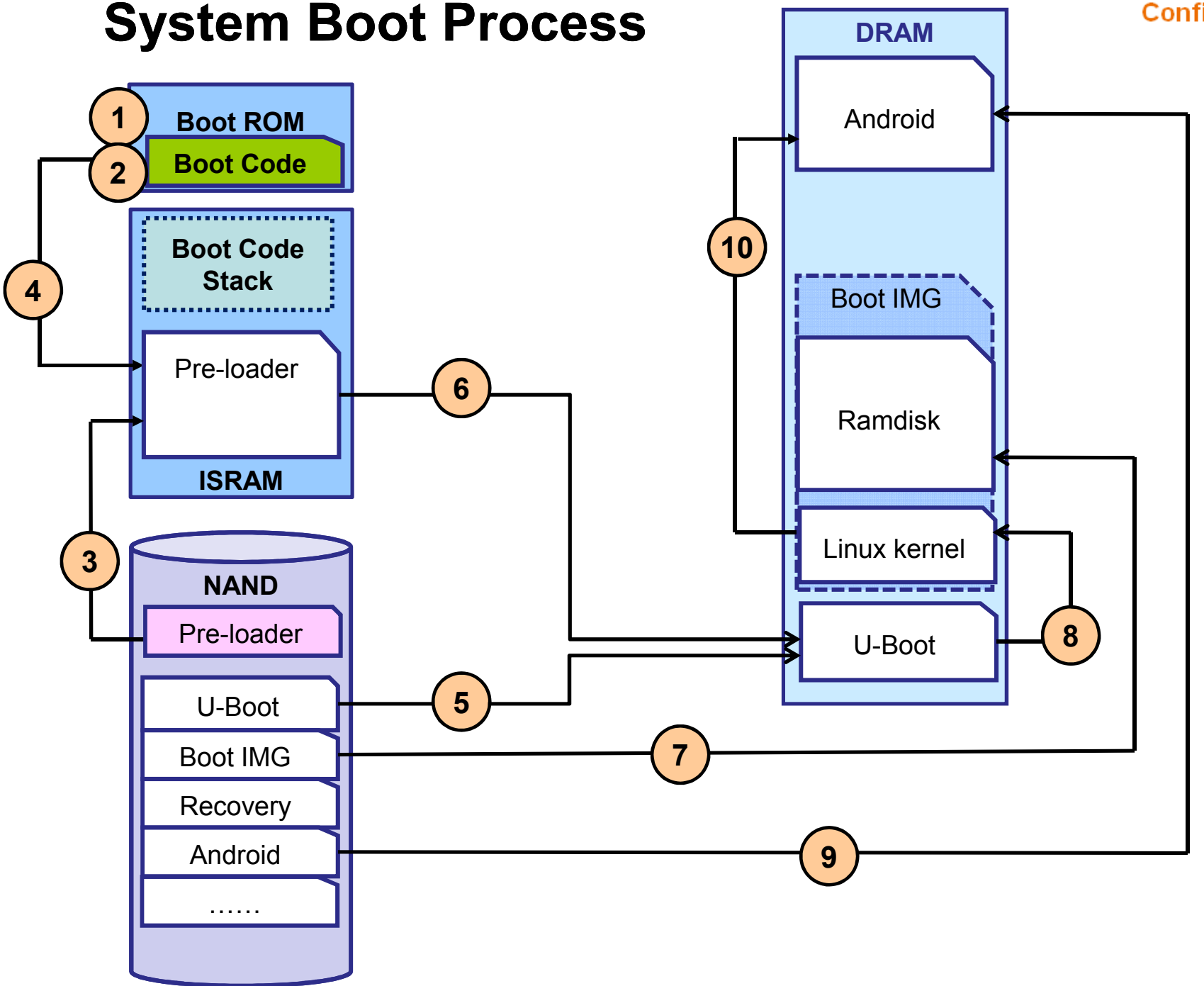
Bootloader Overview

- **Bootloader contains Pre-loader (Initial program Loader) U-Boot (Secondary Loader)**
  - **Pre-loader (MTK in-house developed loader)**
    - takes charge of all the platform dependency work (including initializing EMI / PLL ..) and USB download process.
  - **U-Boot (GPL licensed loader)**
    - prepares the Linux compatible environment (e.g. Linux Kernel Parameter) before entering Linux Kernel.

System Download Process



System Boot Process

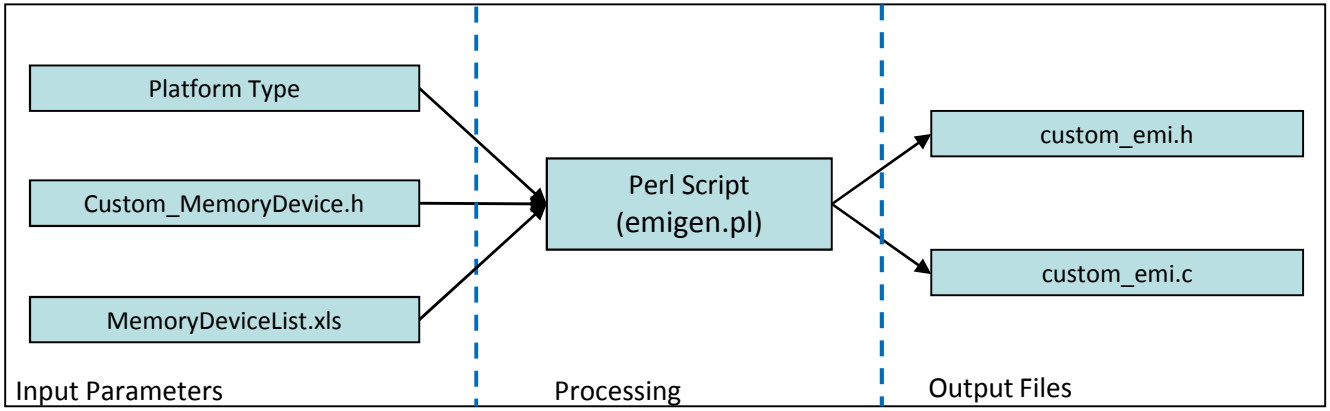


Pre-loader Customization

Custom Feature	Custom File	Detailed Description
EMI (DRAM) Initialization	*preloader\custom\custom\inc\custom_emi.h* *preloader\custom\custom\inc\mt6516_emi.h* *preloader\custom\custom\custom_emi.c* *preloader\custom\custom\MTK_Loader_Info_v4.tag*	No need to modify these files directly, Please modify custom_memorydevice.h Please refer to “Memory” section
NAND Flash Initialization	*preloader\custom\custom\inc\mt6516_nfi.h*	Please refer to “Memory” section
NAND Partition Map (Image Layout)	*preloader\custom\custom\inc\mt6516_partition.h* *preloader\custom\custom\mt6516_partition.c*	Please refer to “Memory” section

EMI Customization

- Introduction
  - Perl script is used to auto generate source file and header file of DDR initialization.
    - Location: alps\tools\emigen\emigen.pl
  - Memory DB file
    - Location: alps\tools\emigen\MemoryDeviceList.xls
    - Please update this file when more memory devices have been verified.



EMI Customization

- Change file list

File	Description
Alps/mediatek/custom/${project_name}/preloader/inc/	
Custom_MemoryDevice.h	The customization file for EMI setting

- Customization item
  - Custom_MemoryDevice.h
  - Eg.,support 3 types MCP (Maxim to 30)
    - #define BOARD_ID XXXX
    - #define CS0_PART_NUMBER[0] <part number in excel>
    - #define CS1_PART_NUMBER[0] <part number in excel>
    - #define CS0_PART_NUMBER[1] <part number in excel>
    - #define CS0_PART_NUMBER[2] <part number in excel>
    - #define CS1_PART_NUMBER[2] <part number in excel>
  - Use ./makeMtk [project_name] emigen to generate emi files.
- **Remind: Must run ETT procedure**

What is Combo MCP Feature

- Collect EMI settings of specified MCP devices into code base when compile time.
- Select correct EMI settings of one MCP device by detected NAND ID in run time.
- NAND ID should be unique between specified MCP devices.
- User can change MCP device without re-compiling / downloading pre-loader image if required MCP devices have already been specified in configure files.

Design Change – MemoryDeviceList.xls

- Location: alps/mtk/tools/emigen/MemoryDeviceList.xls
- Add two columns
  - Board ID
  - NAND ID ((manufactory id + device id)
- If memory device is not MCP, the NAND ID should be empty.

Before

Vendor	Part Number	Density (Mb)	EMI_GENA_VAL	EMI_CONI_VAL
Samsung	K524G2GACB_A050	2048	0x0000030A	0x00314040
Samsung	K524G2GACB_A050_v2	2048	0x0000030A	0x00314060
ELPIDA	EDK1432CABH60	1024	0x0000030A	0x00314000

After

Vendor	Part Number	Density (Mb)	Board ID	NAND ID	EMI_GENA_VAL	EMI_CONI_VAL
Samsung	K524G2GACB_A050	2048	E1KV1	0xBAEC	0x0000030A	0x00314040
Samsung	K524G2GACB_A050_v2	2048	E1KV2	0xBAEC	0x0000030A	0x00314060
ELPIDA	EDK1432CABH60	1024	EVB		0x0000030A	0x00314000

EMI Customization-with combo mcp

- Change file list

File	Description
alps\mtk\src\custom\${BOARD}\preloader\custom\custom\ inc\	
Custom_MemoryDevice.h	The customization file for EMI setting

- Customization item
  - Custom_MemoryDevice.h

```
#define MEMORY_DEVICE_TYPE     LPSDRAM
// Only CS0 or CS1 is allowed for LPSDRAM!
#define BOARD_ID                   E1Kv2

#define CS0_PART_NUMBER[0]         MT29C4G48MANLCJA_6IT
#define CS1_PART_NUMBER[0]         MT29C4G48MANLCJA_6IT

#define CS0_PART_NUMBER[1]         K522F1GACM_A060
#define CS1_PART_NUMBER[1]         K522F1GACM_A060
```

  - Use ./makeMtk [project] emigen to generate emi files.
- **Remind: Must run ETT procedure**

EMI Customization-Limitation

- NAND ID must be unique, not support MCPs that NAND IDs are the same but MCP part numbers are different.
- If mounted memory device is not MCP
  - It means DRAM part number must be specified, not support the auto-detect feature by NAND ID.
  - NAND ID of corresponding record in MemoryDeviceList.xls must be empty.
  - There must be only one set of part number defined in custom_MemoryDevice.h to select EMI settings.
- Pre-loader size limitation
  - Pre-loader size will be checked in compile time.
  - Due to pre-loader size limitation, currently only support 15 MCP records in MT6516.

U-Boot Customization

Custom Feature	Custom File	Detailed Description
Power-off Charging	*Alps/mediatek/custom/${project_name}/uboot/cust_battery.h*	Please refer to “Battery” section
LCM Display (Boot LOGO)	*Alps/mediatek/custom/${project_name}/uboot/inc/cust_display.h* *Alps/mediatek/custom/${project_name}/uboot/cust_display.c*	Please refer to “Display” section
Key Mapping	*Alps/mediatek/custom/${project_name}/uboot/inc/configs/${project}.h*	Recovery Key/ Factory key configuration.
NAND Flash Initialization	*Alps/mediatek/custom/${project_name}/uboot/inc/cust_nand.h*	Please refer to “Memory” section
NAND Partition Map (Image Layout)	*Alps/mediatek/custom/${project_name}/uboot/inc/mt65xx_partition.h* *Alps/mediatek/custom/${project_name}/uboot/mt65xx_partition.c*	Please refer to “Memory” section
LED	*Alps/mediatek/custom/${project_name}/uboot/cust_leds.c*	Please refer to “NLED” section

Key Mapping - Uboot

- **In u-boot, we can modify the file**
  - *Alps/mediatek/custom/${project_name}/uboot/inc/configs/${project_name}.h*

```
#define MT65XX_RECOVERY_KEY     1       /* KEY_VOLUMEUP */
#define MT65XX_FACTORY_KEY      2       /* KEY_VOLUMEDOWN */
```

- **How to get the key number for the keys?**
  - *Alps/mediatek/custom/${project_name}/kernel/dct/dct/cust_kpd.h*

Pre-loader/Uboot/Kernel

- In pre-loader/uboot/kernel, we can modify the file
  - *Alps/mediatek/custom/${project_name}/preloader/inc/mt65xx_nfi.h*
  - *Alps/mediatek/custom/${project_name}/uboot/inc/cust_nand.h*
  - *Alps/mediatek/custom/${project_name}/kernel/core/src/board-custom.h*
- NFI configurations can be set in this file
- Modify this file according to the NAND MCP configurations
  - If can not find the MCP in the table, please contact to MTK side.

```
static const flashdev_info g_FlashTable[]={
    //micro
    {0xAA2C,  5,  8,  256,  128,  2048,  0x01113,  "MT29F2G08ABD",  0},
    {0xB12C,  4,  16, 128,  128,  2048,  0x01113,  "MT29F1G16ABC",  0},
    {0xBA2C,  5,  16, 256,  128,  2048,  0x01113,  "MT29F2G16ABD",  0},
    {0xAC2C,  5,  8,  512,  128,  2048,  0x01113,  "MT29F4G08ABC",  0},
    //samsung
    {0xBAEC,  5,  16, 256,  128,  2048,  0x01123,  "K522H1GACE",    0},
    {0xBCEC,  5,  16, 512,  128,  2048,  0x01123,  "K524G2GACB",    0},
    {0xDAEC,  5,  8,  256,  128,  2048,  0x33222,  "K9F2G08U0A",    RAMDOM_READ},
    {0xF1EC,  4,  8,  128,  128,  2048,  0x01123,  "K9F1G08U0A",    RAMDOM_READ},
    {0xAAEC,  5,  8,  256,  128,  2048,  0x01123,  "K9F2G08R0A",    0},
    //hynix
    {0xD3AD,  5,  8,  1024, 256,  2048,  0x44333,  "HY27UT088G2A",  0},
    {0xA1AD,  4,  8,  128,  128,  2048,  0x01123,  "H8BCS0PJ0MCP",  0},
    {0xBCAD,  5,  16, 512,  128,  2048,  0x01123,  "H8BCS0UN0MCR",  0},
    {0xBAAD,  5,  16, 256,  128,  2048,  0x01123,  "H8BCS0SN0MCR",  0},
```

NAND Partition Layouts

- In MTK's Android system, we have 13 different NAND partitions in the whole system
- Pre-loader
  - Pre-loader image
  - Handles all the download and secure boot procedures
- U-boot
  - Second loader image
  - Handles most hardware initializations and bring-up entire Linux kernel
- Logo
  - Boot-up logo image
- Bootimg
  - Linux kernel image and it's root file system

NAND Partition Layouts

- Recovery
  - Recovery kernel image and it's root file system
  - Handles all the system recovery and firmware update functionalities
- System
  - Android system image
- NvRam
  - Stores the hardware related information, such as calibration data, MAC address, IMEI … etc.
- Cache
  - For Android internal used
  - Store Android internal cache data or web cache data
- Misc
  - Used for the recovery procedure (power loss)

NAND Partition Layouts

- User
  - Used for Android system to store user data such as user contacts, settings, installed applications … etc.
- SECCFG
  - Reserved for the security platform used
- EXPDB
  - Reserved.

Android Partition Layout

Index	Partition	Type	Start	End	Size	Size (KB)	Size2	Size(HEX)	DL
1	PRELOADER	Raw data	0	20000	128 KB	128	131072	20000	1
2	NVRAM	YAFFS2	20000	320000	3 MB	3072	3145728	300000	0
3	SECCFG	Raw data	320000	340000	128 KB	128	131072	20000	0
4	UBOOT	Raw data	340000	3A0000	384 KB	384	393216	60000	1
5	BOOTIMG	Raw data	3A0000	9A0000	6 MB	6144	6291456	600000	1
6	RECOVERY	Raw data	9A0000	FA0000	6 MB	6144	6291456	600000	1
7	SEC_RO	YAFFS2	FA0000	10C0000	1.125 M	1152	1179648	120000	1
8	MISC	Raw data	10C0000	1120000	384KB	384	393216	60000	0
9	ANDROID	YAFFS2	1120000	8920000	120 MB	122880	125829120	7800000	1
10	CACHE	YAFFS2	8920000	C520000	60 MB	61440	62914560	3C00000	0
11	LOGO	Raw data	C520000	C820000	3 MB	3072	3145728	300000	1
12	EXPDB	Raw data	C820000	C8C0000	640 KB	640	655360	A0000	0
13	USRDATA	YAFFS2	C8C0000	END	128KB	128		20000	1
14	END	Raw data	0	0	0	0	200.75	0	0

NAND Partition Layout Customizations

- If we need to change the partition layout, please make sure the settings in the pre-loader, u-boot and kernel are correct
- The path we need to modify
  - Pre-loader
    - *Alps/mediatek/custom/${project_name}/preloader/inc/mt65xx_partition.h*
    - *Alps/mediatek/custom/${project_name}/preloader/mt65xx_partition.c*
  - U-boot
    - *Alps/mediatek/custom/${project_name}/uboot/inc/mt65xx_partition.h*
    - *Alps/mediatek/custom/${project_name}/uboot/mt65xx_partition.c*
  - Kernel
    - *Alps/mediatek/custom/${project_name}/kernel/core/src/partition.h*
- It must be checked that the settings in theses files are consistent
- Partitions size should be enough
- Don’t forget to modify Scatterfile

Outline

Connecting

- WIFI
- MT6620 Customization

Sensor System

- Sensor Hal
- Sensor Driver Customization

Audio

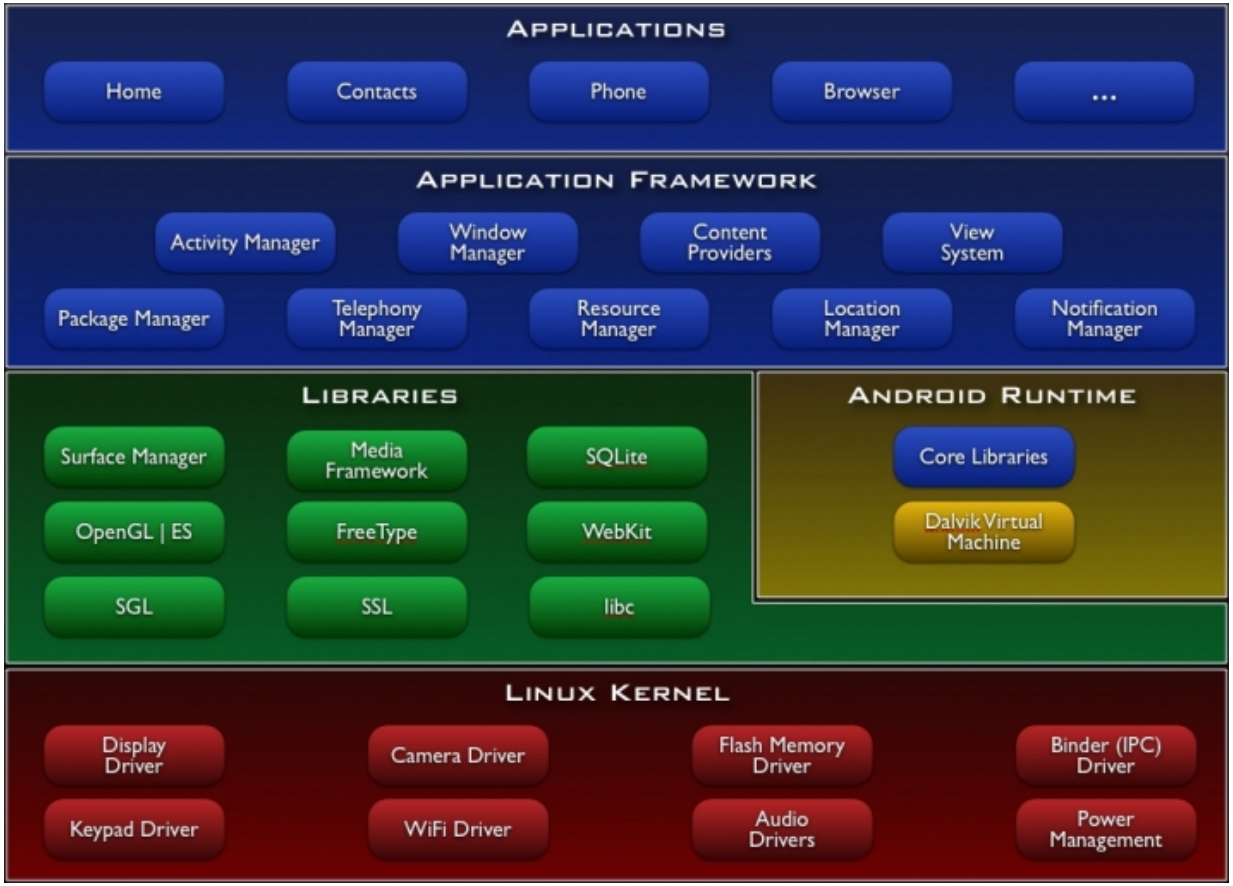
Misc.

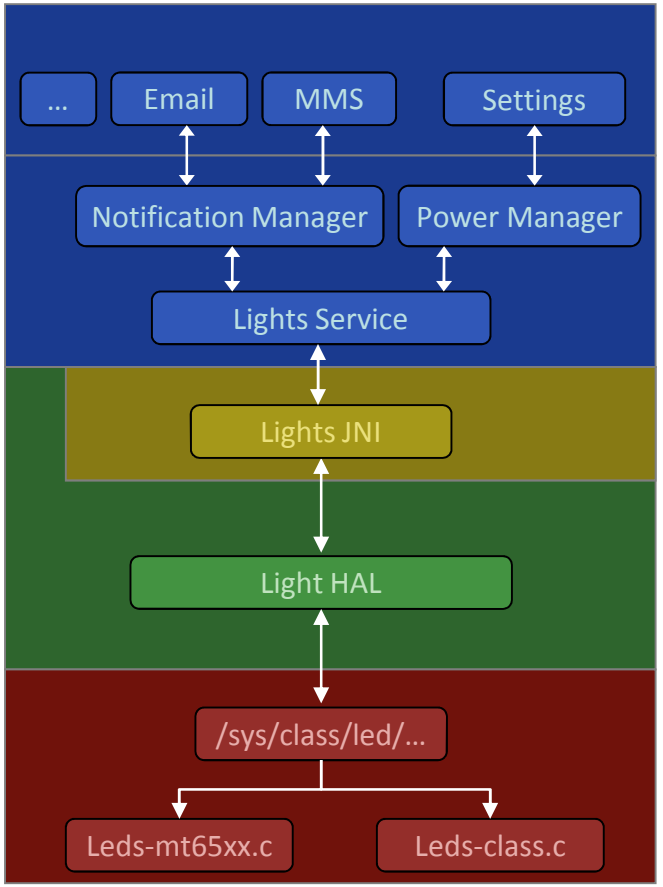
- GPIO
- I2C
- PMIC
- Headset

Appendix

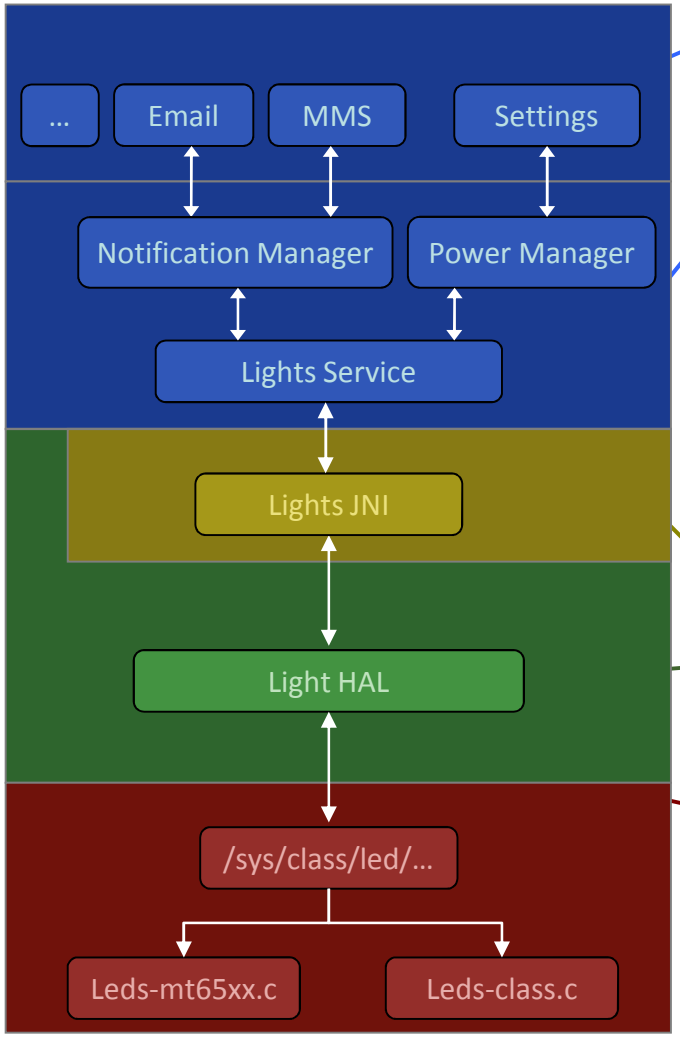
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

Android Lights





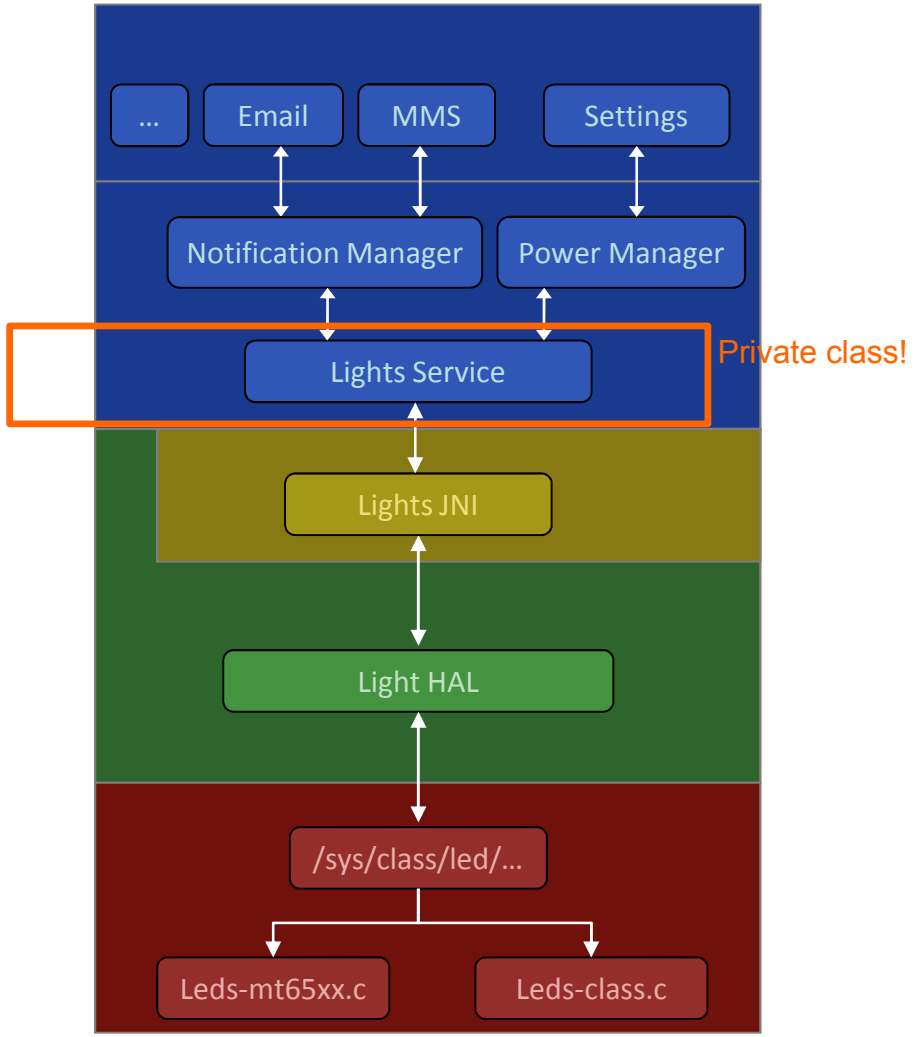
Lights Architecture



- Request system to control lights
- Notification Manager
  - Notifications
  - Battery
  - Attetion
- Power Manger
  - Backlight
  - KeyBoard
  - Buttons
- Lights Service
- The connection between Java Layer and Native HAL
- Hardware Abstraction Layer
  - A wrapper of device-dependent implementation
- /sys/class/led/...
  - Usr space interface through sys files
- Leds-mt65xx.c(android2.2), leds.c(android2.3)
  - kernel driver
- Leds-class.c
  - Class register driver

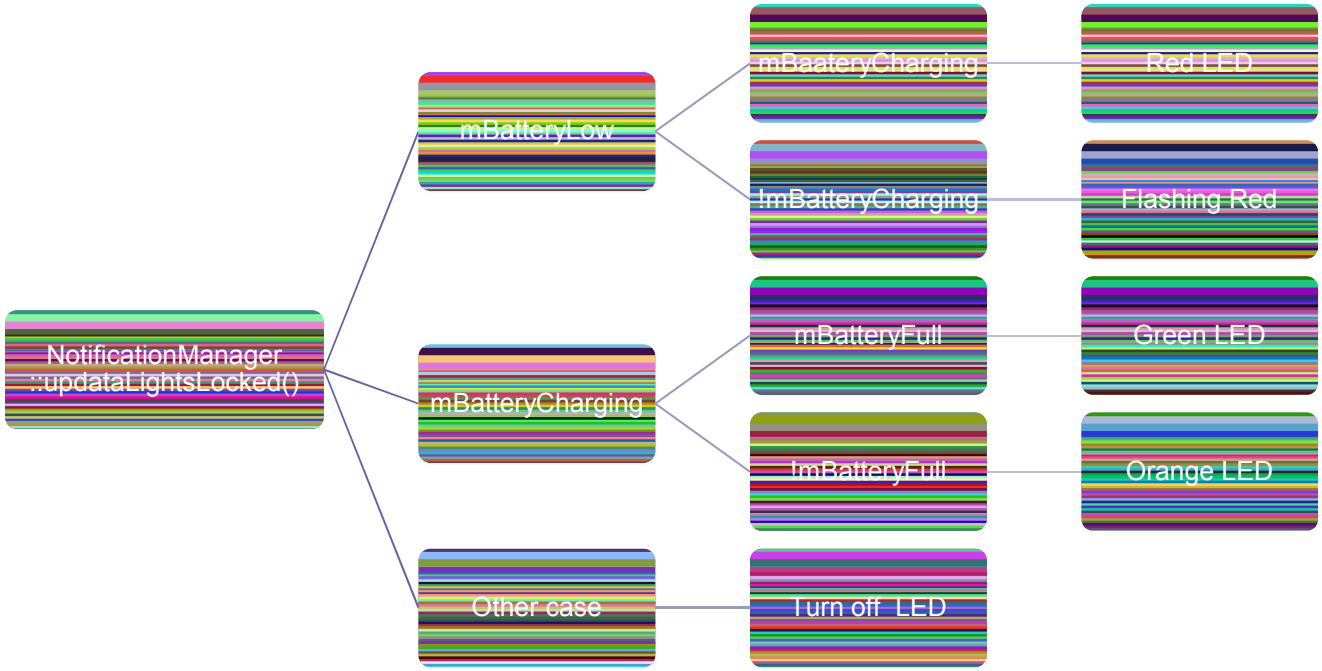
Lights Service Introduce

- Android lights service supports eight lights types:
  - backlight
  - Keyboard
  - Button
  - Battery
  - Notifications
  - Attention
  - Bluetooth
  - WIFI
- backlight supports 256 level brightness
- Leds supports ARGB format



Charging LED

- NotificationManagerService
  - Battery lights :



  - Notification light
  - Attention light

Following pictures show Red, Green and Orange LED light separately.

Lights HAL

- Lights.c(hardware/mtk/liblights/)
  - Led-mt65xx.c or leds.c(kernel/driver/leds/)
  - Led-class.c(kernel/driver/leds/)
  - Sysfiles: /sys/class/leds/
  - Provide a mechanism for communication between kernel space and user space

```
/
 |-- sys
     |--class
        |--leds
           |--red
           |--green
           |--blue
           |--jogball-backlight
           |--keyboard-backlight
           |--button-backlight
           |--lcd-baclight
```

Backlight Customization

- Customization item – Overview
  - Cust_leds_def.h

```
#define CUST_LEDS_BACKLIGHT_PMIC_PARA
```

  - Cust_leds.c

```
Static int custom_backligh_func(kal_bool state)
{…}

/* An customization example: button and LCD backlight controlled by PMIC */
static struct cust_mt65xx_led cust_led_list[MT65XX_LED_TYPE_TOTAL] = {
          {"red",               MT65XX_LED_MODE_PWM,   1},
          {"green",             MT65XX_LED_MODE_GPIO, 10},
          {"blue",              MT65XX_LED_MODE_NONE, -1},
          {"jogball-backlight", MT65XX_LED_MODE_NONE, -1},
         {"keyboard-backlight",MT65XX_LED_MODE_CUST, (int)custom_backligh_func},
          {"button-backlight",  MT65XX_LED_MODE_PMIC, MT65XX_LED_PMIC_BUTTON},
          {"lcd-backlight",     MT65XX_LED_MODE_PMIC, MT65XX_LED_PMIC_LCD},
};
```

- Customization files for kernel and uboot are independent of each other

Backlight Customization

- Customization item
  - **name**: The name of Android Light
    - The string and the sequence in the list MUST not be modified.
  - **mode**: The backlight control mode
    - MT65XX_LED_MODE_NONE: This light type is not supported
    - MT65XX_LED_MODE_PWM: Control by PWM
    - MT65XX_LED_MODE_PMIC: Control by PMIC
    - MT65XX_LED_MODE_CUST: Control by user-defined function
  - **data**
    - For MT65XX_LED_MODE_NONE mode
      - Ignore value
    - For MT65XX_LED_MODE_PWM mode
      - PWM number (0~6)
    - For MT65XX_LED_MODE_PMIC mode
      - MT65XX_LED_PMIC_KEYBOARD
      - MT65XX_LED_PMIC_BUTTON
      - MT65XX_LED_PMIC_LCD
    - For MT65XX_LED_MODE_CUST mode
      - **cust_brightness_set** function pointer

Brightness mapping

- If use PMIC to control the LCD backlight
  - **Should change the mapping function**
    - The level from APP is from 0~255
    - Should mapping to PMIC level (0~31)

```
unsigned int brightness_mapping(unsigned int level)
{
    level = level < 8?1:level/8;
    return level;
}
```

Outline

Overview
- Chip
- Android
- Build Command

Boot up
- Bootloader
- Android Booting

Lights System

Battery Manager
- ***Battery Service***
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

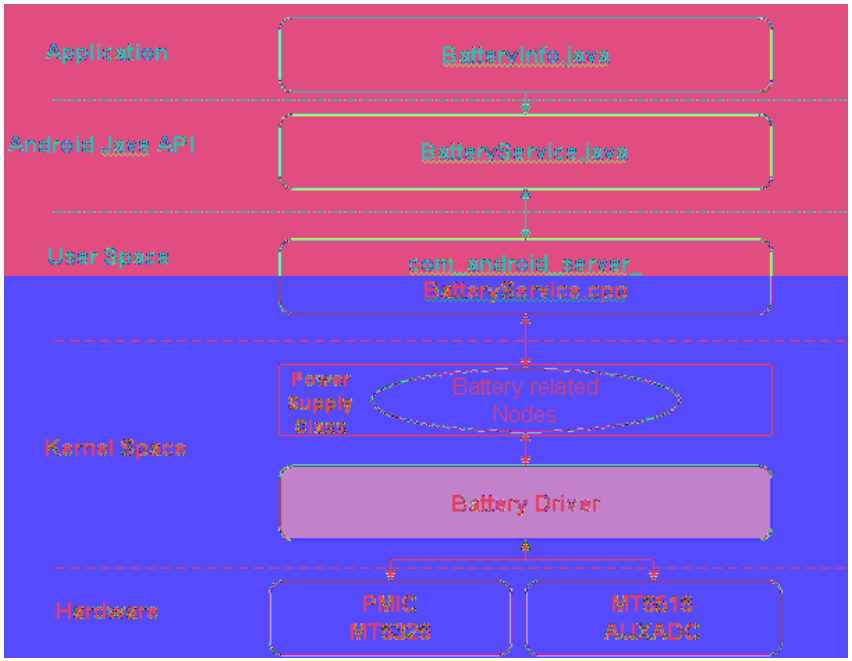
Audio

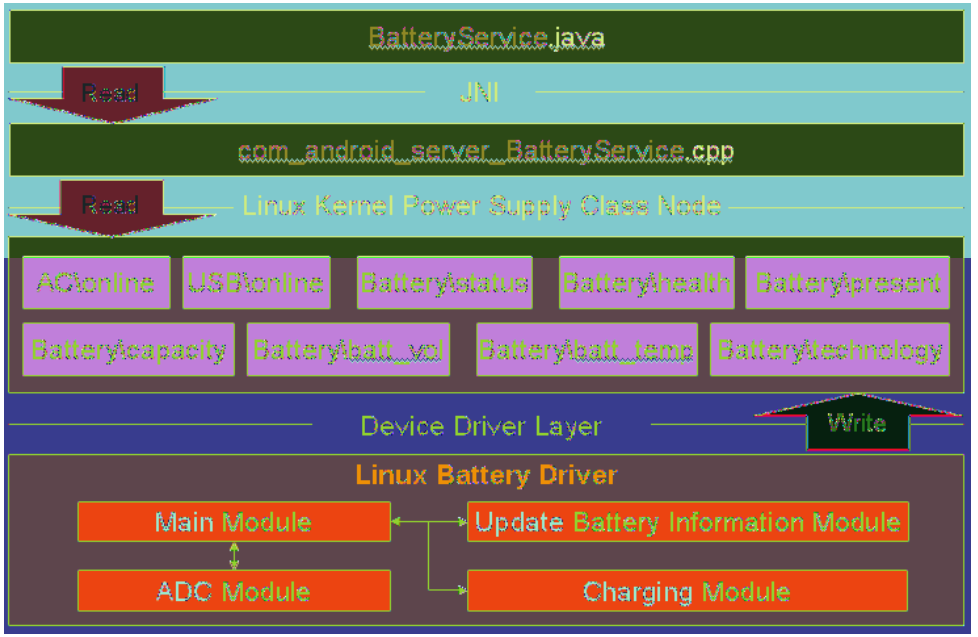
Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

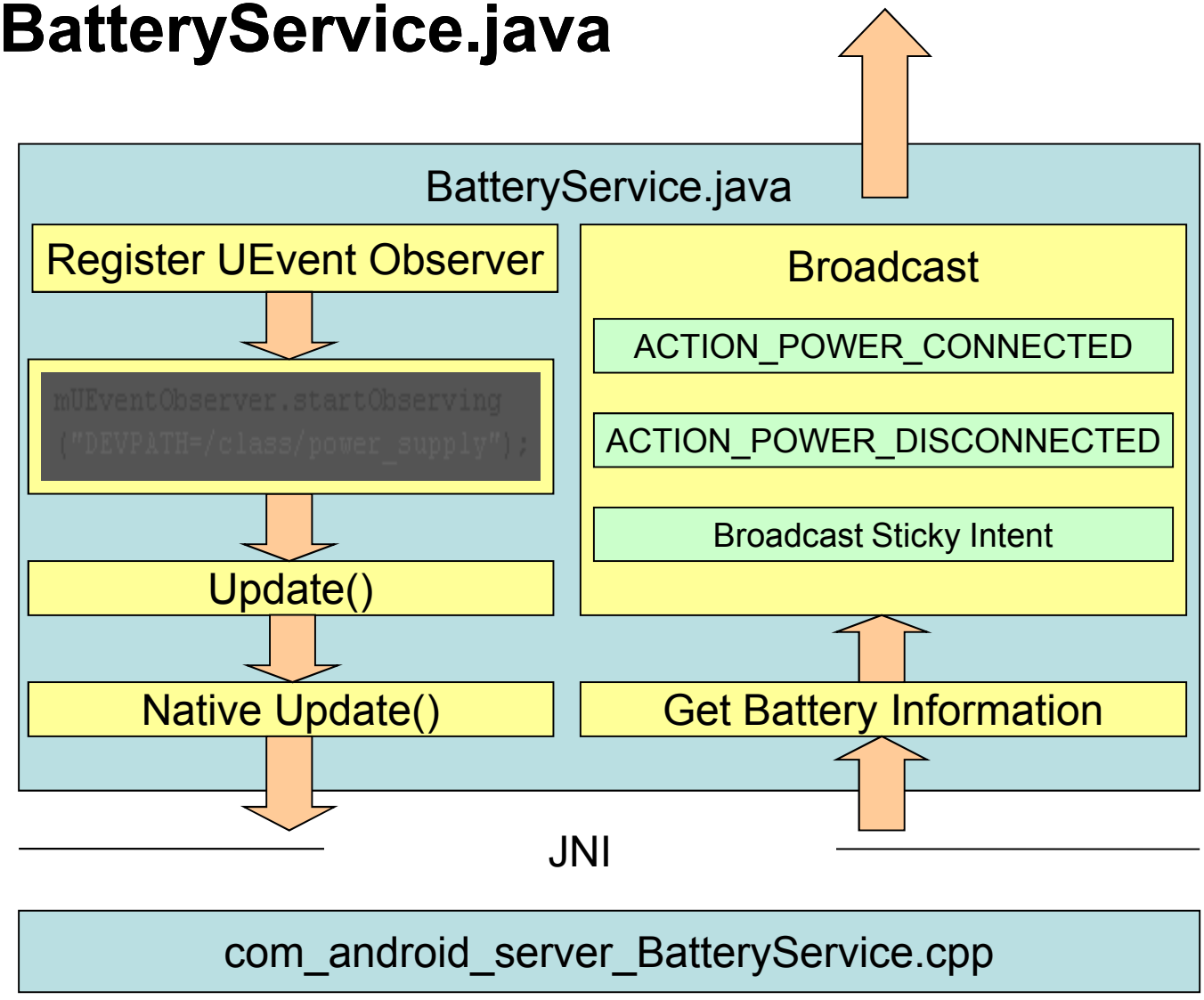
Battery Introduction

- Introduction

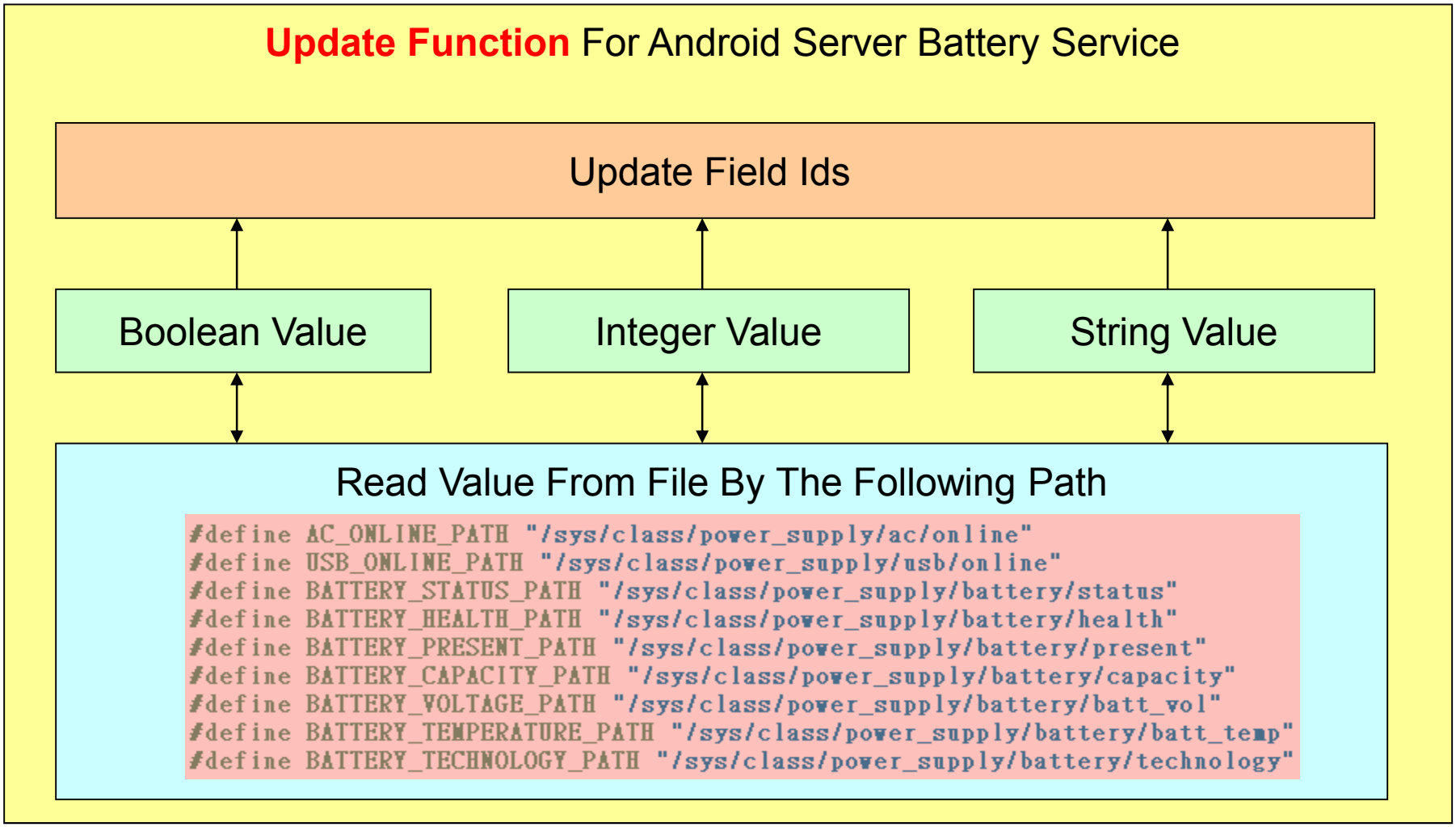




BatteryService.java

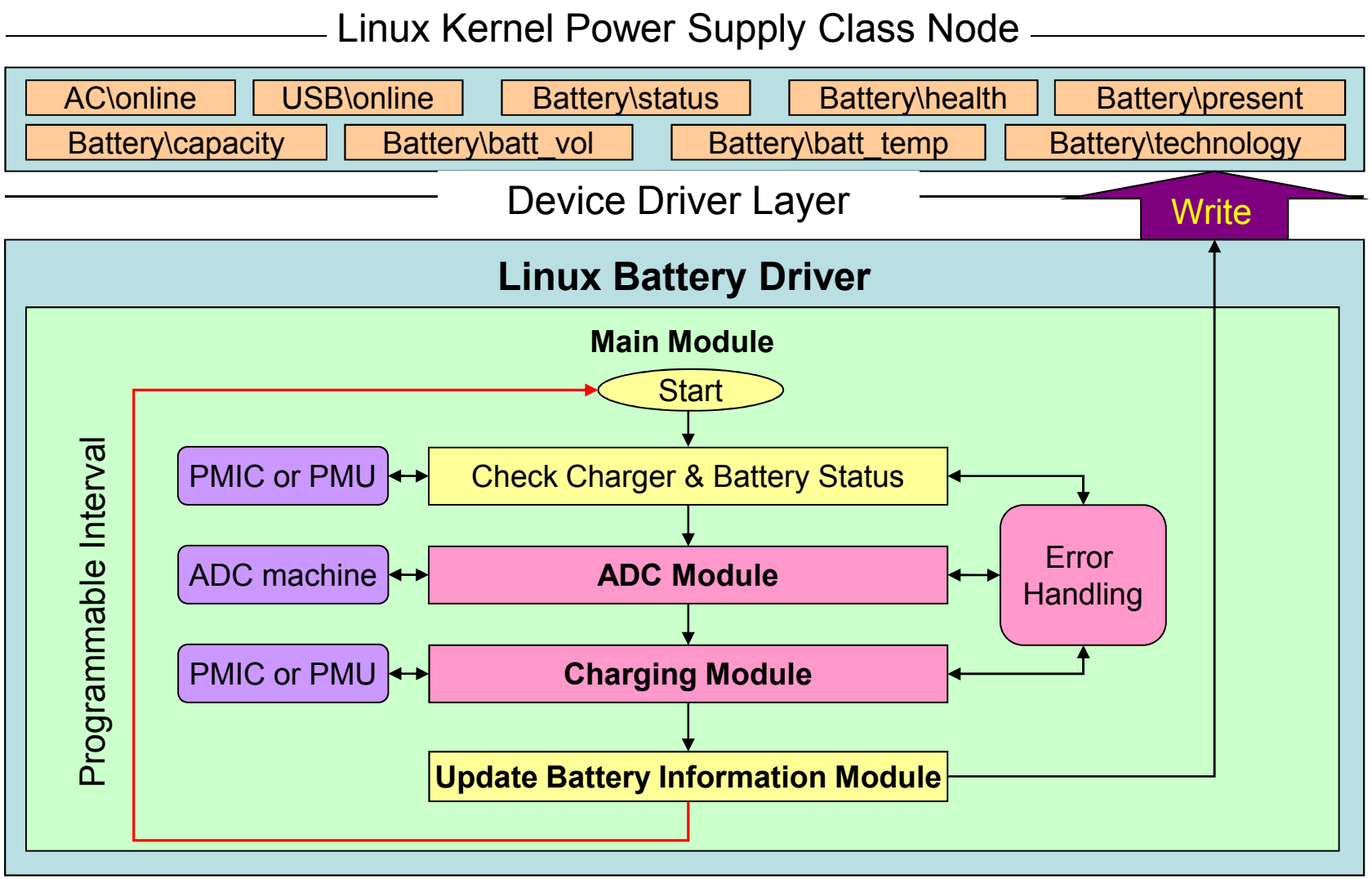


Battery Information Update Function



Working Module

Hareware / Register
Unique module
Common module



Reason : This is a simple, common and extendable working module for all BU.

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- ***Battery Charging***
- Power Off Charging

HCIT
- Keypad
- Touch
- Display

Connecting
- WIFI

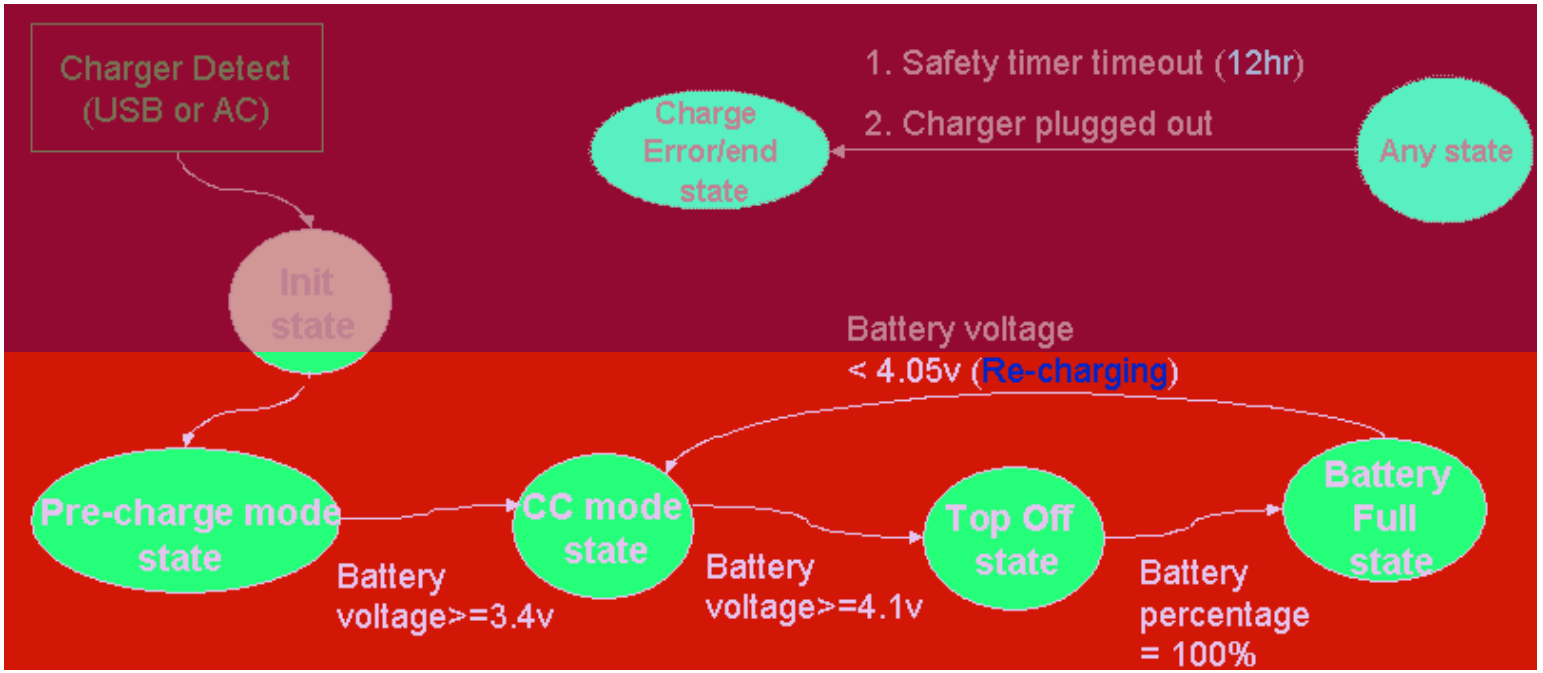
Sensor System
- Sensor Hal
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- SDHC
- Camera
- Factory
- Jogball
- GPS
- LCM Porting
- Recovery
- USB
- BT
- RTC
- Mdoem
- FM

Battery Charging State Machine



Battery Customization

- Change file list

File	Description
\alps\mediatek\platform\mt6573\kernel\drivers\power	
Mt6573_battery.c	The implementation of battery charging related APIs.
\alps\mediatek\platform\mt6573\kernel\drivers\power	
mt6573_battery.h	The battery charging related settings (internal).
\alps\mediatek\custom\${project_name}\kernel\battery\battery	
cust_battery.h	The battery charging related settings (customer).

- Customization item

Customization Item		Description	Default Value	Range
Battery Voltage-to-Percentage Table		11 level battery percentage (0,10,20,30,40,50,60,70,80,90,100%)	By battery SPEC.	3400mV (0%) ~ 4200mV (100%)
Normal Charging Current	USB	CC mode USB charging current	400mA	800 / 700 / 600 / 500 / 400 / 300 / 200 / 100 mA
	AC	CC mode AC charging current	600mA	
USB-IF Charging Current	USB	CC mode USB charging current (USB suspended)	0mA	800 / 700 / 600 / 500 / 400 / 300 / 200 / 100 mA
		CC mode USB charging current (USB un-configured)	1000mA	
		CC mode USB charging current (USB configured)	400mA	
	AC	CC mode AC charging current	600mA	
Recharging Battery Voltage		Recharging for keeping the battery capacity	4050mV	3900 ~ 4150mV
Battery Temperature Charging Protection	Higher	Above the hi-temperature ➔ disable charging	45C	- 20 ~ 60 C
	Lower	Below the low-temperature ➔ disable charging	0C	
Charging Resister		The resister for measuring the charging current	2 (=0.2Ohm)	
Battery Sense Resister		The resister for measuring the battery sense voltage	2	
I Sense Resister		The resister for measuring the I sense voltage	2	
Charger Sense Resister		The resister for measuring the charger sense voltage	5	
V_CHARGER_MAX		The max value of charger voltage	6000 (mV)	> V_CHARGER_MIN
V_CHARGER_MIN		The min value of charger voltage	4400 (mV)	< V_CHARGER_MAX
V_CHARGER_ENABLE		Enable/disable the charger voltage protection	0	(1:ON, 0:OFF)
RBAT_PULL_UP_R		The pull up resister for measuring battery temperature	24000 (Ohm)	
RBAT_PULL_UP_VOLT		The pull up voltage for measuring battery temperature	2500 (mV)	
TBAT_OVER_CRITICAL_LOW		The extreme value for calculating resister	68237	Suggest to fix this value
BAT_TEMP_PROTECT_ENABLE		Enable/disable the battery temperature protection	0	(1:ON, 0:OFF)

Runtime Battery Logging Entry

- Path
  - cd /proc
- Enable logging
  - echo 1 > batdrv_log
- Disable logging
  - echo 0 > batdrv_log
- Demo

```
# echo 1 > batdrv_log
enable battery driver log system
# [BATTERY] LOG ----------------------------------------------------------------
[BATTERY:ADC] VCHR:5011 BAT_SENSE:4166 I_SENSE:4276 Current:550
[BATTERY:AVG] Temp:25 vbat:4090 batsen:4166 SOC:92 state:1002 chrdet:1 chrtype:1 Vchrin:50
11 Icharging:207
[BATTERY] CC mode charge, timer=100 !!
[BATTERY] STANDARD_HOST CC mode charging : 5
[BATTERY] LOG ----------------------------------------------------------------
[BATTERY:ADC] VCHR:4996 BAT_SENSE:4166 I_SENSE:4276 Current:550
[BATTERY:AVG] Temp:25 vbat:4092 batsen:4166 SOC:93 state:1002 chrdet:1 chrtype:1 Vchrin:49
96 Icharging:214
[BATTERY] CC mode charge, timer=110 !!
[BATTERY] STANDARD_HOST CC mode charging : 5

# echo 0 > batdrv_log
Disable battery driver log system
#
```

LOG Analysis

- BATTERY:ADC
  - VCHR, BAT_SENCE, I_SENCE
    - 当下从ADC channel抓出来的数值
  - Current
    - 从BAT_SENCE, I_SENCE换算出来的数值
  - 存在10%的误差

- BATTERY:AVG
  - vbat, Icharging, SOC (battery percentage)
    - vbat : 10分钟内BAT_SENCE平均的数值
    - Icharging : 10分钟内I_SENCE平均的数值
    - SOC : 10分钟内 battery percentage平均的数值
  - state
    - CHR_CC (0x1002), CHR_CV (0x1003), CHR_BATFULL (0x1004)
    - CHR_ERROR (0x1005)
  - chrtype
    - STANDARD_HOST (1), CHARGING_HOST (2),NONSTANDARD_CHARGER (3), STANDARD_CHARGER(4)

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- ***Power Off Charging***

HCIT
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor Hal
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

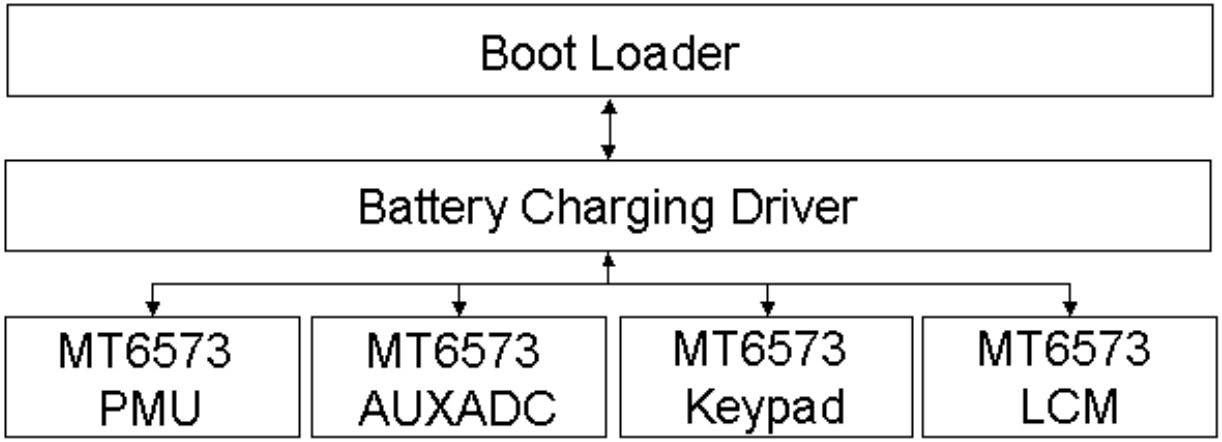
Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix

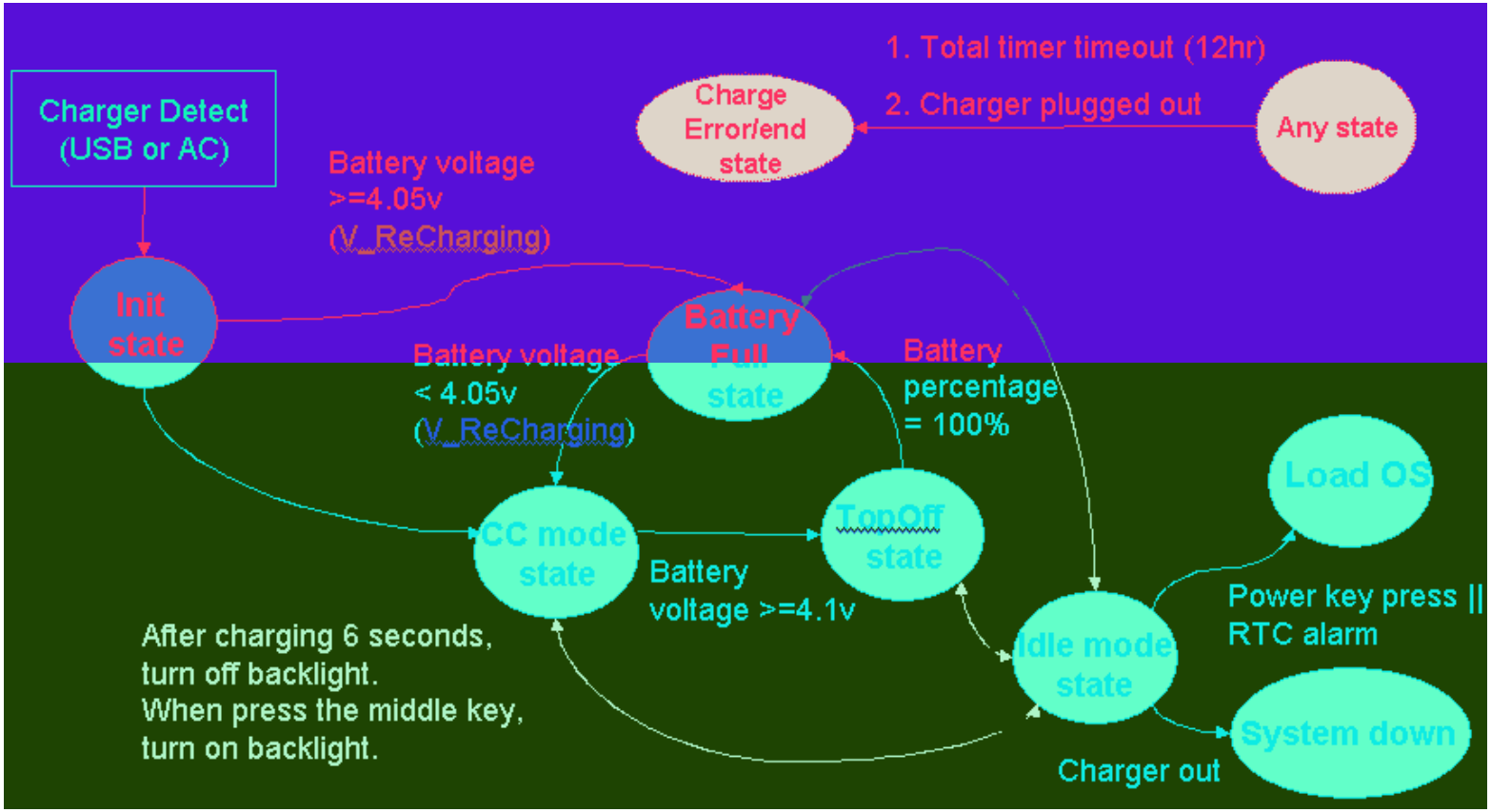
• OFN	Jogball	USB	FM
• SDHC	GPS	BT	
• Camera	LCM Porting	RTC	
• Factory	Recovery	Mdoem	

Power Off Charging Introduction

- Introduction



Power Off Charging State Machine



Power Off Charging Customization

- Change file list

File	Description
alps\mediatek\platform\mt6573\uboot	
mt6573_bat.c	The implementation of battery charging related APIs.
alps\mediatek\custom\${project_name}\uboot\inc	
cust_battery.h	The battery charging related settings.

- Customization item

Customization Item		Description	Default Value	Range
Battery Voltage-to-Percentage Table		11 level battery percentage (0,10,20,30,40,50,60,70,80,90,100%)	By battery SPEC.	3400mV (0%) ~ 4200mV (100%)
Normal Charging Current	USB	CC mode USB charging current	400mA	800 / 700 / 600 / 500 / 400 / 300 / 200 / 100 mA
	AC	CC mode AC charging current	600mA	
USB-IF Charging Current	USB	CC mode USB charging current (USB suspended)	0mA	800 / 700 / 600 / 500 / 400 / 300 / 200 / 100 mA
		CC mode USB charging current (USB un-configured)	1000mA	
		CC mode USB charging current (USB configured)	400mA	
	AC	CC mode AC charging current	600mA	
Recharging Battery Voltage		Recharging for keeping the battery capacity	4050mV	3900 ~ 4150mV
Battery Temperature Charging Protection	Higher	Above the hi-temperature ➔ disable charging	45C	- 20 ~ 60 C
	Lower	Below the low-temperature ➔ disable charging	0C	
Charging Resister		The resister for measuring the charging current	2 (=0.2Ohm)	
Battery Sense Resister		The resister for measuring the battery sense voltage	2	
I Sense Resister		The resister for measuring the I sense voltage	2	
Charger Sense Resister		The resister for measuring the charger sense voltage	5	
V_CHARGER_MAX		The max value of charger voltage	6000 (mV)	> V_CHARGER_MIN
V_CHARGER_MIN		The min value of charger voltage	4400 (mV)	< V_CHARGER_MAX
V_CHARGER_ENABLE		Enable/disable the charger voltage protection	0	(1:ON, 0:OFF)
RBAT_PULL_UP_R		The pull up resister for measuring battery temperature	24000 (Ohm)	
RBAT_PULL_UP_VOLT		The pull up voltage for measuring battery temperature	2500 (mV)	
TBAT_OVER_CRITICAL_LOW		The extreme value for calculating resister	68237	Suggest to fix this value
BAT_TEMP_PROTECT_ENABLE		Enable/disable the battery temperature protection	0	(1:ON, 0:OFF)

Wake up the backscreen

- Set CHARGING_IDLE_MODE to 1
  - alps\mediatek\custom\${project_name}\uboot\inc\cust_battery.h
  - Uboot will turn off the backlight after some seconds in power off charging

- BL_SWITCH_TIMEOUT
  - Define the timeout time, default is 6 seconds
  - check BAT_CheckBatteryStatus () in alps\mediatek\platform\mt6573\uboot\mt6573_bat.c
  - Press the **BACKLIGHT_KEY** to wake up the back screen

```
if (mt6573_detect_key(BACKLIGHT_KEY)) {
    bl_switch = KAL_FALSE;
    bl_switch_timer = 0;
    printf("[BATTERY] mt65xx_backlight_on\r\n");
}
```

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- ***Keypad***
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

Audio

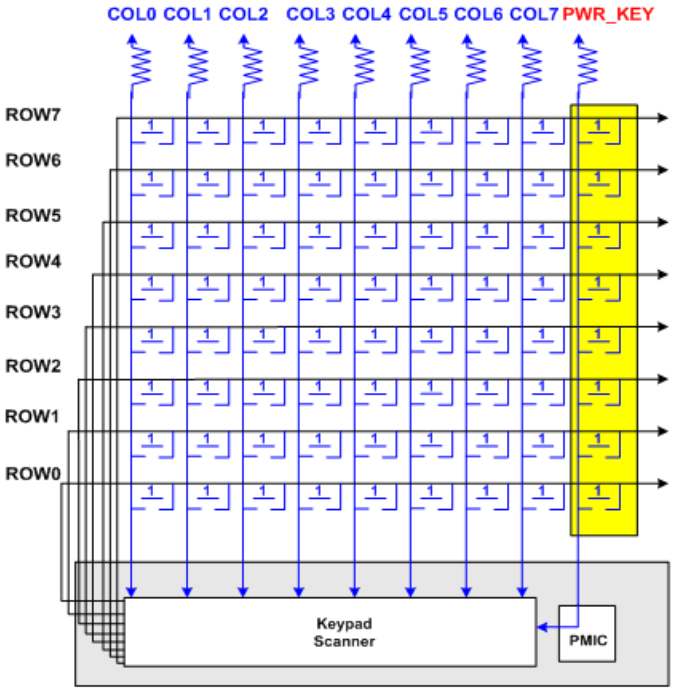
Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

Keypad HW

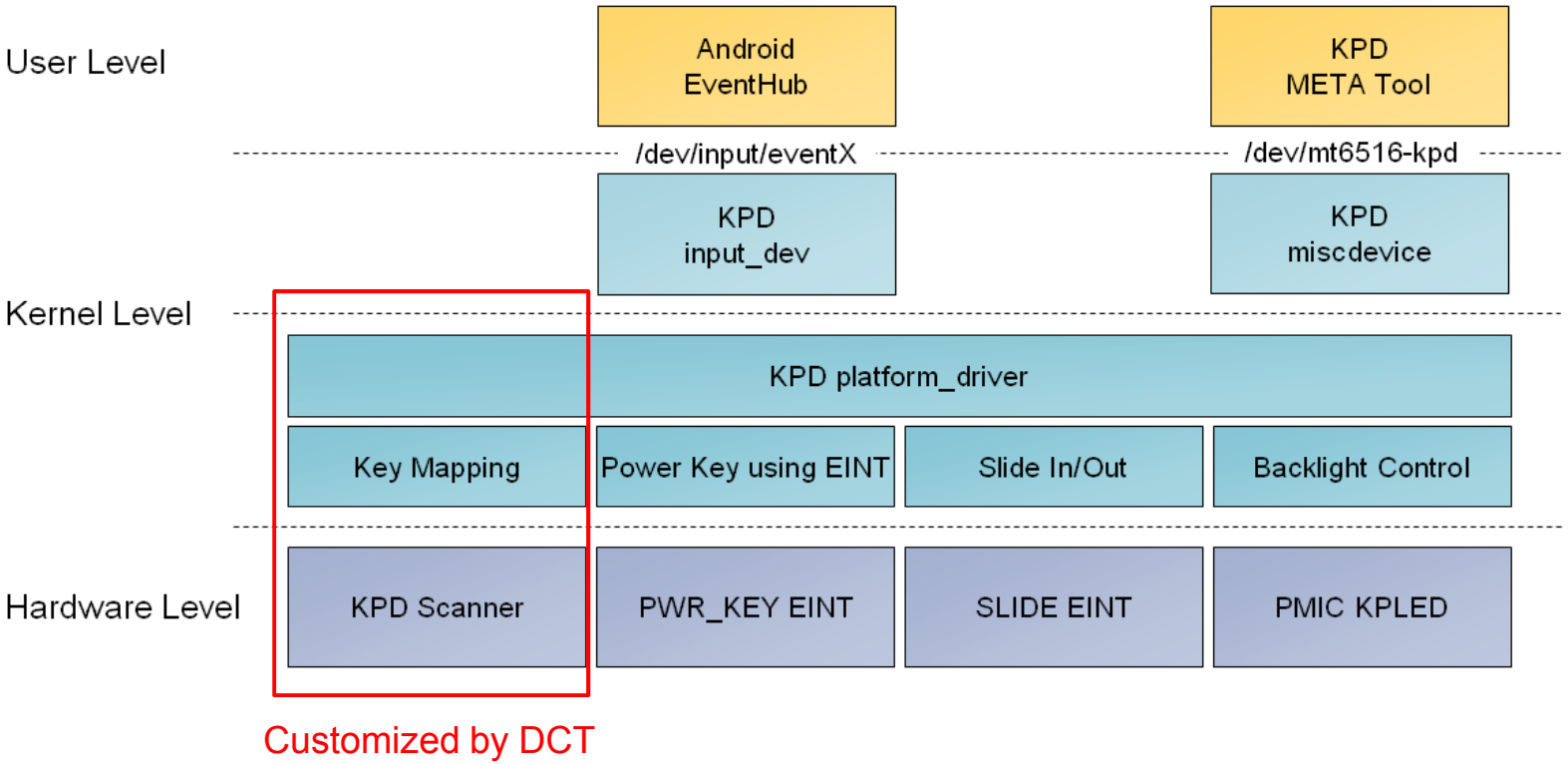
- Keypad matrix has 8 rows and (8 + 1) columns
  - HW keycode are 0 to 71
- Column 8 (HW keycode are 8, 17, 26, 35, 44, 53, 62, 71) is dedicated for Power key, so we can have 8 * 8 = 64 keys detected through Keypad matrix

	COL0	COL1	COL2	COL3	COL4	COL5	COL6	COL7	PWRKEY
ROW7	63	64	65	66	67	68	69	70	71
ROW6	54	55	56	57	58	59	60	61	62
ROW5	45	46	47	48	49	50	51	52	53
ROW4	36	37	38	39	40	41	42	43	44
ROW3	27	28	29	30	31	32	33	34	35
ROW2	18	19	20	21	22	23	24	25	26
ROW1	9	10	11	12	13	14	15	16	17
ROW0	0	1	2	3	4	5	6	7	8



Keypad Driver

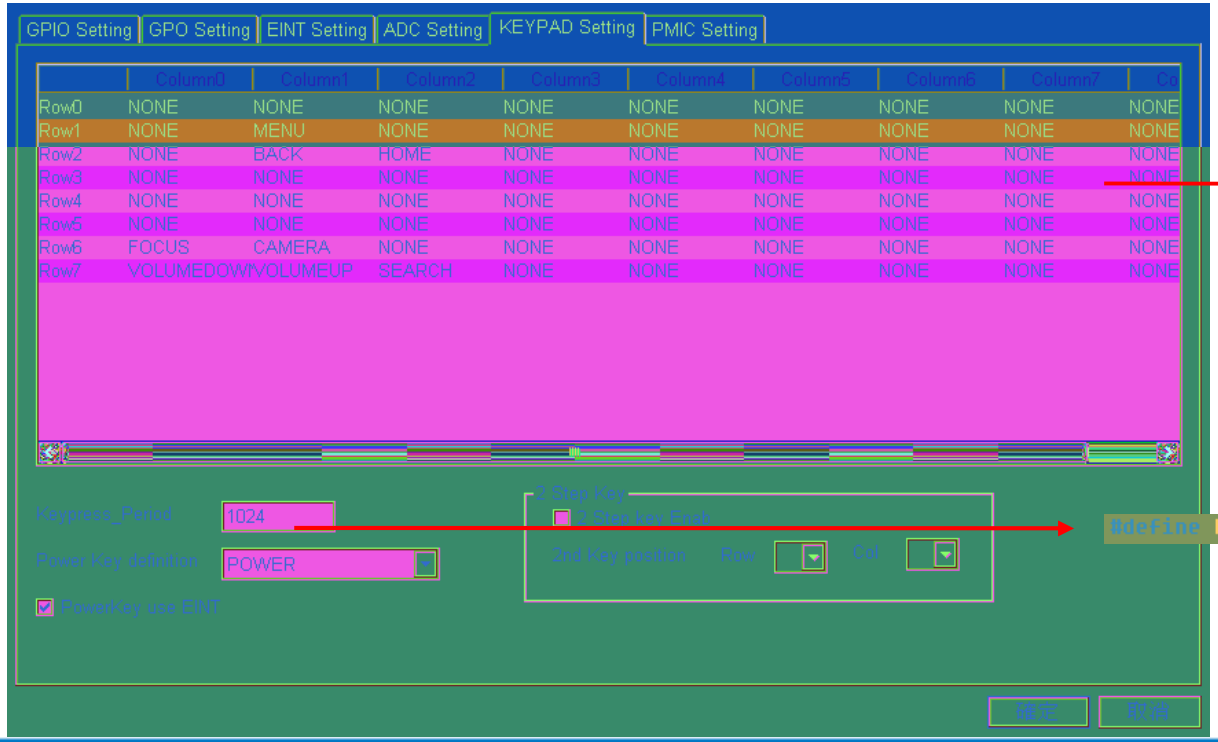
- The architecture of Keypad driver is shown below



DCT (Driver Customization Tool)

- DCT is used to customize the key mapping and de-bounce time
  - Key Mapping: it is corresponding to Keypad matrix
  - De-Bounce Time: the number in “Keypress_Period” divided by 32 means the Keypad’s de-bounce time (millisecond)

◎ alps/mediatek/source/dct/DrvGen.exe

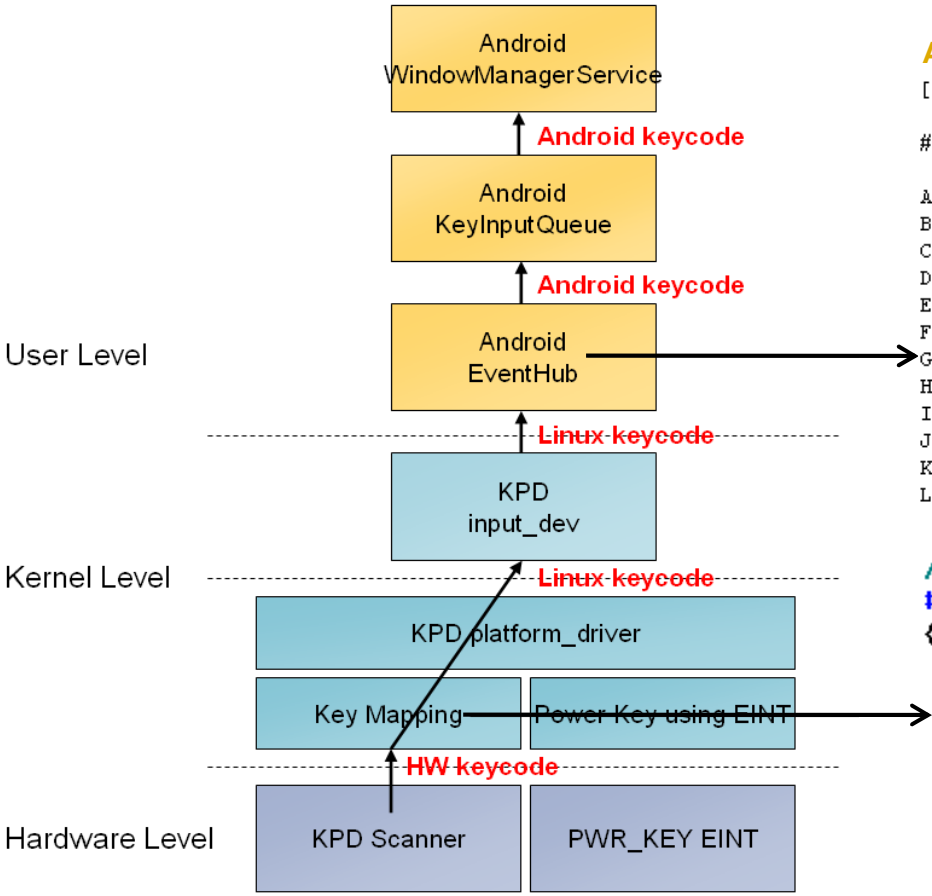


◎ alps/mediatek/custom/${BOARD}/kernel/dct/dct/cust_kpd.h

```
/* HW keycode [0 ~ 71] -> Linux keycode */
#define KPD_INIT_KEYMAP()       \
{       \
        [10] = KEY_MENU,                \
        [19] = KEY_BACK,                \
        [20] = KEY_HOME,                \
        [54] = KEY_FOCUS,               \
        [55] = KEY_CAMERA,              \
        [63] = KEY_VOLUMEDOWN,          \
        [64] = KEY_VOLUMEUP,            \
        [65] = KEY_SEARCH,              \
}
```

```
#define KPD_KEY_DEBOUNCE  1024      /* (val / 32) ms */
```

Key Report flow



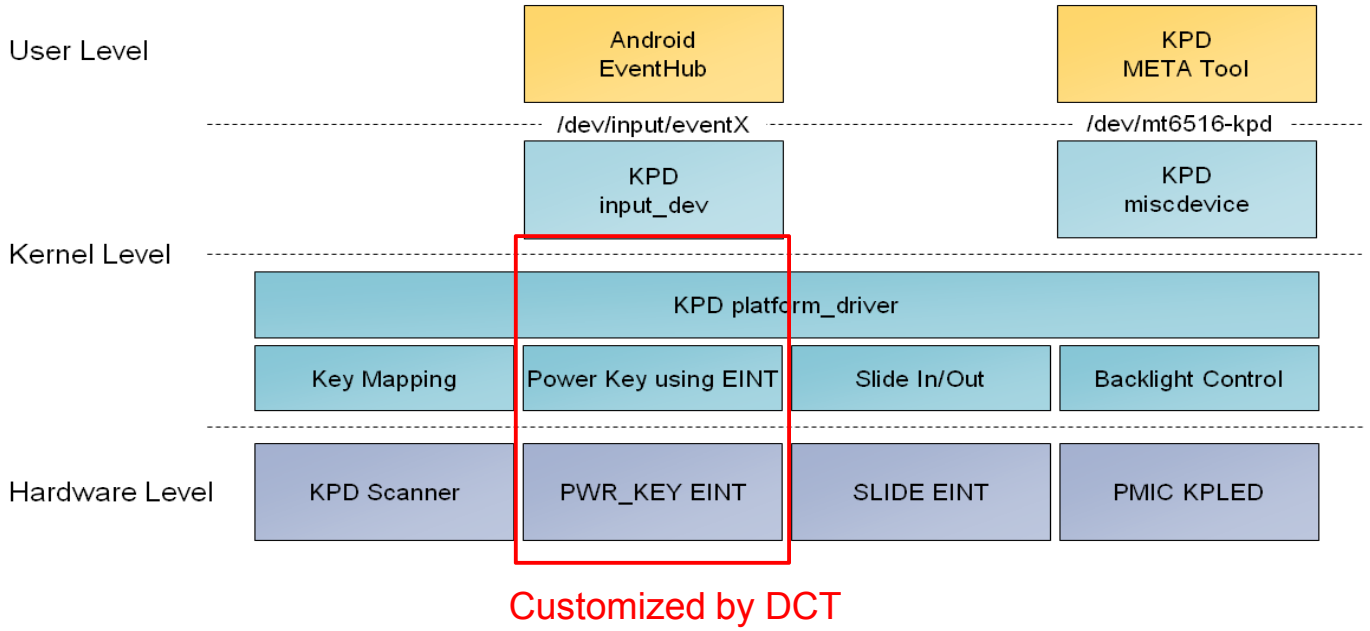
Alps/mediatek/config/$(Project_name)/mt6573-kpd.kl

```
[type=QWERTY]

# keycode       display number  base    caps    fn      caps_fn

A               'A'     '%'     'a'     'A'     '%'     0x00
B               'B'     '='     'b'     'B'     '='     0x00
C               'C'     '8'     'c'     'C'     '8'     0x00E7
D               'D'     '5'     'd'     'D'     '5'     0x00
E               'E'     '2'     'e'     'E'     '2'     0x0301
F               'F'     '6'     'f'     'F'     '6'     0x00A5
G               'G'     '-'     'g'     'G'     '-'     '_'
H               'H'     '['     'h'     'H'     '['     '{'
I               'I'     '$'     'i'     'I'     '$'     0x0302
J               'J'     ']'     'j'     'J'     ']'     '}'
K               'K'     '"'     'k'     'K'     '"'     '~'
L               'L'     '''     'l'     'L'     '''     '`'
```

```
/* HW keycode [0 ~ 71] -> Linux keycode */
#define KPD_INIT_KEYMAP()       \
{       \
        [10] = KEY_MENU,                \
        [19] = KEY_BACK,                \
        [20] = KEY_HOME,                \
        [54] = KEY_FOCUS,               \
        [55] = KEY_CAMERA,              \
        [63] = KEY_VOLUMEDOWN,          \
        [64] = KEY_VOLUMEUP,            \
        [65] = KEY_SEARCH,              \
}
```

Power Key

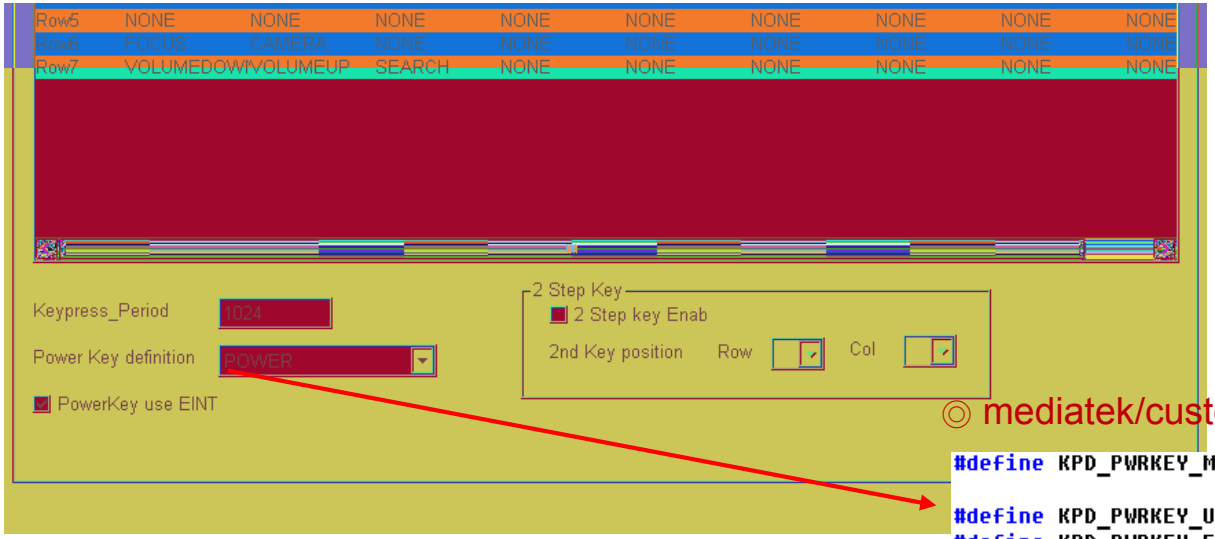
- We use EINT to detect Power key's pressing and releasing
  - In MT6516, Power key uses EINT4 (GPIO63)
  - Power key is strongly suggested to disconnect from Keypad matrix



DCT (Driver Customization Tool)

- DCT can be used to customize the Power key setting
  - Always check the box “PowerKey use EINT”
    - You have to configure EINT related setting by using DCT
  - Choose “POWER” or “ENDCALL” in “Power Key definition” so that Keypad driver can use this mapping to report the key event

◎ mtk/src/dct/DrvGen.exe

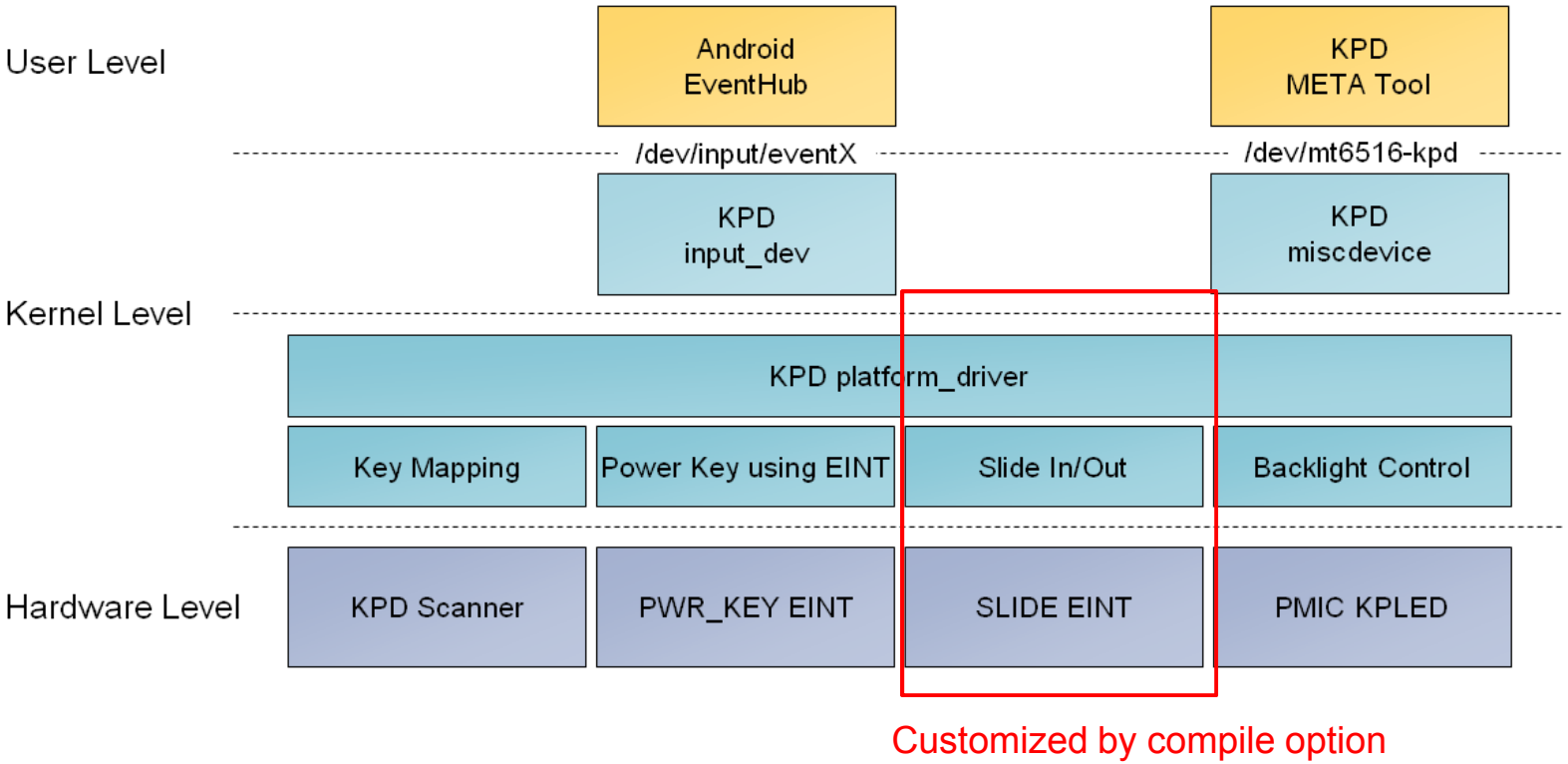


◎ mediatek/custom/${BOARD}/kernel/dct/dct/cust_kpd.h

```
#define KPD_PWRKEY_MAP    KEY_POWER

#define KPD_PWRKEY_USE_EINT       KPD_YES
#define KPD_PWRKEY_EINT           CUST_EINT_KPD_PWRKEY_NUM
#define KPD_PWRKEY_DEBOUNCE       CUST_EINT_KPD_PWRKEY_DEBOUNCE_CN
#define KPD_PWRKEY_POLARITY       CUST_EINT_KPD_PWRKEY_POLARITY
#define KPD_PWRKEY_SENSITIVE      CUST_EINT_KPD_PWRKEY_SENSITIVE
```

Keypad Driver

- The architecture of Keypad driver is shown below



Compile Option

- If the device has Slide QWERTY keypad, you can turn on KPD_HAS_SLIDE_QWERTY to enable the sliding support function
  - We use EINT to detect QWERTY keypad's sliding in and sliding out, so you have to configure EINT related setting by using DCT
- KPD_HAS_SLIDE_QWERTY (default: No)

◎ alps/mediatek/custom/${BOARD}/kernel/dct/dct/cust_kpd.h

```
#define KPD_HAS_SLIDE_QWERTY    KPD_NO
#define KPD_SLIDE_EINT          CUST_EINT_KPD_SLIDE_NUM
#define KPD_SLIDE_DEBOUNCE      CUST_EINT_KPD_SLIDE_DEBOUNCE_CN /* (cn / 32) ms */
#define KPD_SLIDE_POLARITY      CUST_EINT_KPD_SLIDE_POLARITY
#define KPD_SLIDE_SENSITIVE     CUST_EINT_KPD_SLIDE_SENSITIVE
```

Keypad Files

- For Android2.3

File	Description
mtk/src/custom/{project name}/kernel/dct/dct/	
cust_kpd.h	The header file generated by DCT
mtk/src/custom/{project name}/kernel/kpd/kpd/	
mt6516_kpd.h	The header file used to enable/disable some functionality of Keypad driver
mt6516_kpd_bkl.c	Keypad backlight related implementation
kernel/drivers/input/keyboard/	
mt6516_kpd.c	The core of Keypad driver

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- ***Touch***
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem

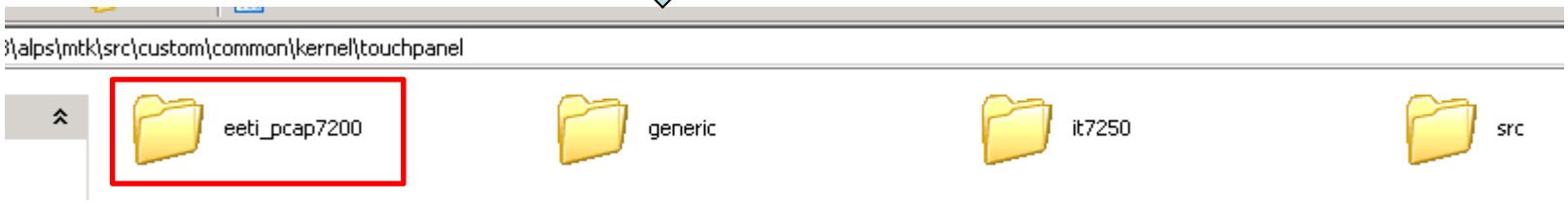
Jogball
SDHC
Camera
Factory

USB
GPS
LCM Porting
Recovery

Touch Panel Driver in ALPS

- **Select TP Driver's type**
  - makefile option in

Alps/mediatek/config/$(project)/ProjectConfig.mk

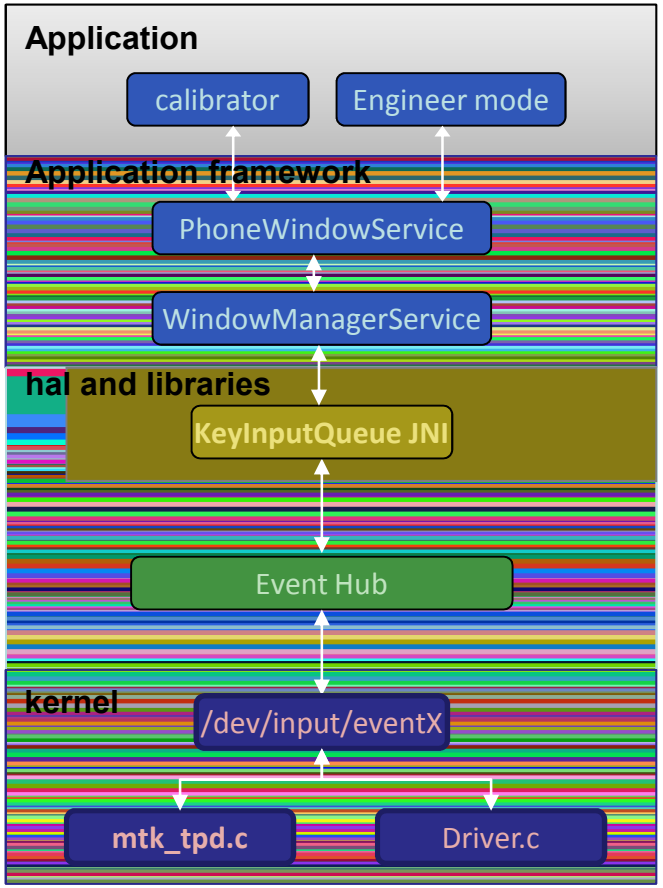


Generic ---- > R touch
Src ---- > R and C touch Common part
Others ---- > C touch

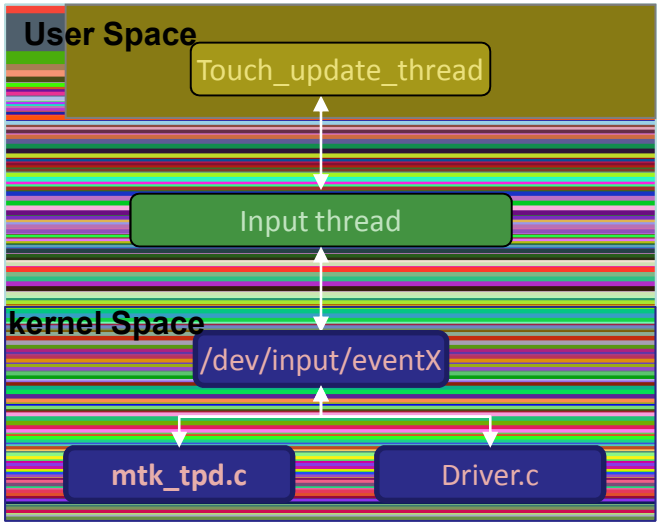
Touch Panel Driver in ALPS

➢ Architecture

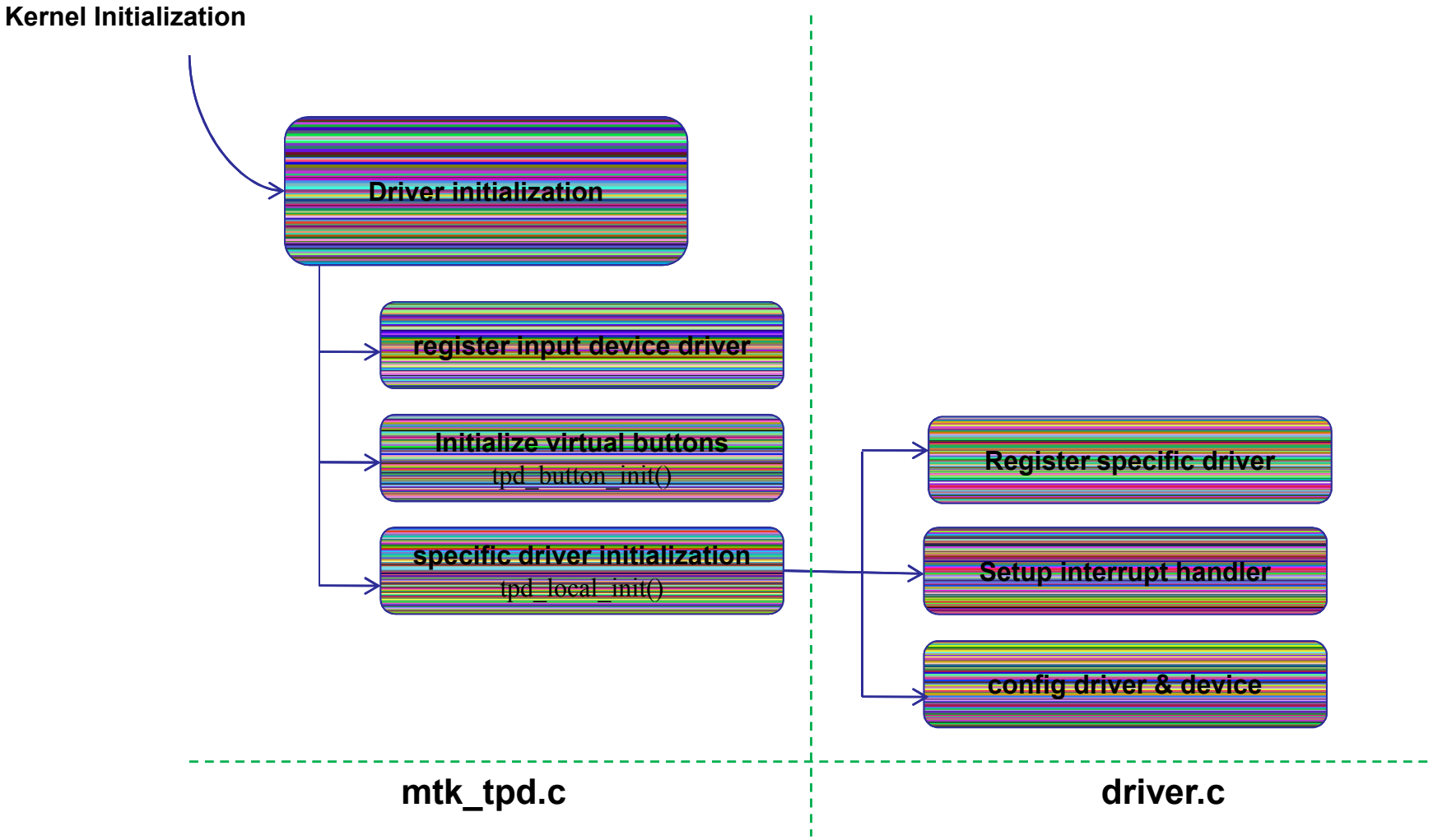
Normal Mode



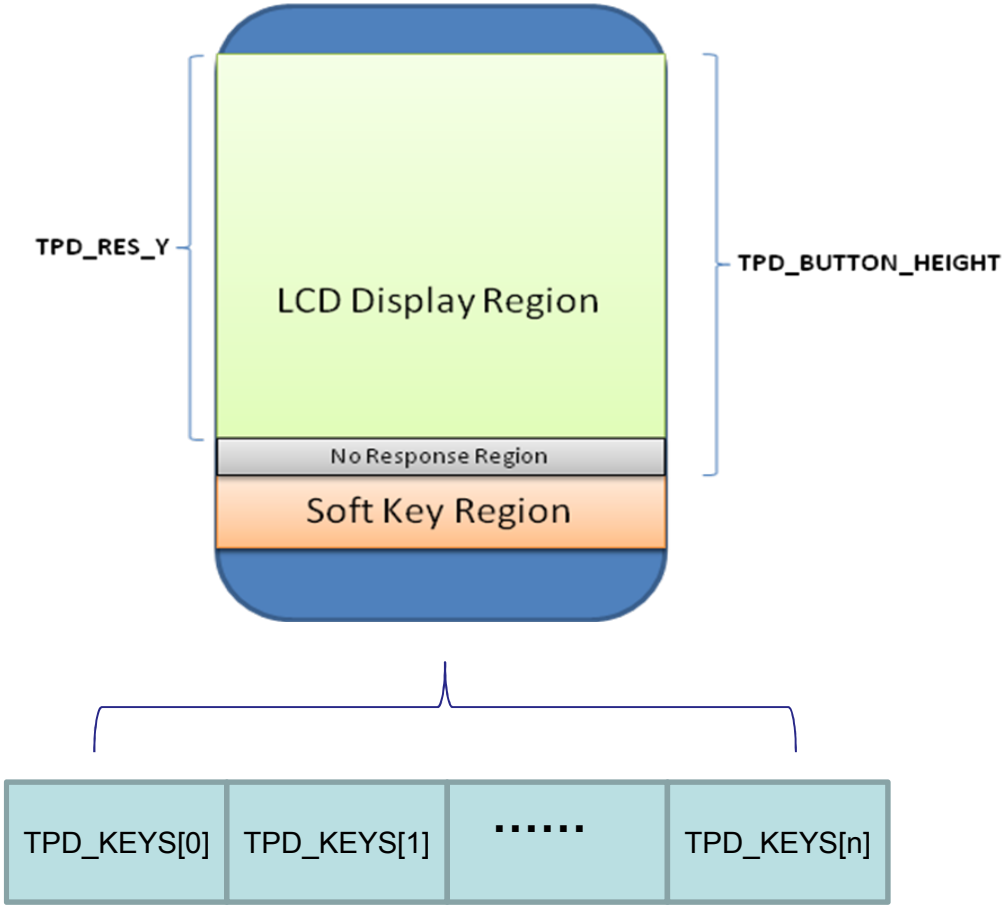
Factory Mode



Touch Panel Driver Flow – Initialization



Button Related Customization



Button Related Customization

category	name	type	description
Button Related	TPD_HAVE_BUTTON	Any	If virtual button is needed to be implemented by touch panel driver, define this macro.
	TPD_CUSTOM_BUTTON	Any	If button layout is different with predefined ones, this macro should be defined and the function **tpd_button** should be implemented
	TPD_BUTTON_HEIGHT	int	It defines the actual y coordinate of touch panel where soft key should be recognized.
	TPD_KEY_COUNT	int	Defines the number of soft key
	TPD_KEYS	int array	Defines the key code of each soft key.

File Name	Location
Tpd_custom_xx.h	Android 2.2 alps\mtk\src\custom\${BOARD}\kernel\touchpanel\${touch folder}\ Android 2.3 Alps\mediatek\custom\[project]\kernel\touchpanel\${touch folder}\

Calibration Related Customization

category	name	type	description
Calibration	TPD_HAVE_CALIBRATION	any	If it is defined, touch panel calibration functionality will be turned on
	TPD_CALIBRATION_MATRIX	int array	It's an 8 elements integer array. It defined the default calibration matrix for touch panel driver.
	TPD_CUSTOM_CALIBRATION	any	If it's needed to implement customized calibration function, define this macro and implement tpd_calibrate() function.
	TPD_WARP_START	int array	These two macros should be defined as an integer array with 4 elements. They should be both defined to enable calibration warp around edge. When they are defined, warp algorithm will be applied to defined edge region.
	TPD_WARP_END	int array	

R-type Touch Panel Customization

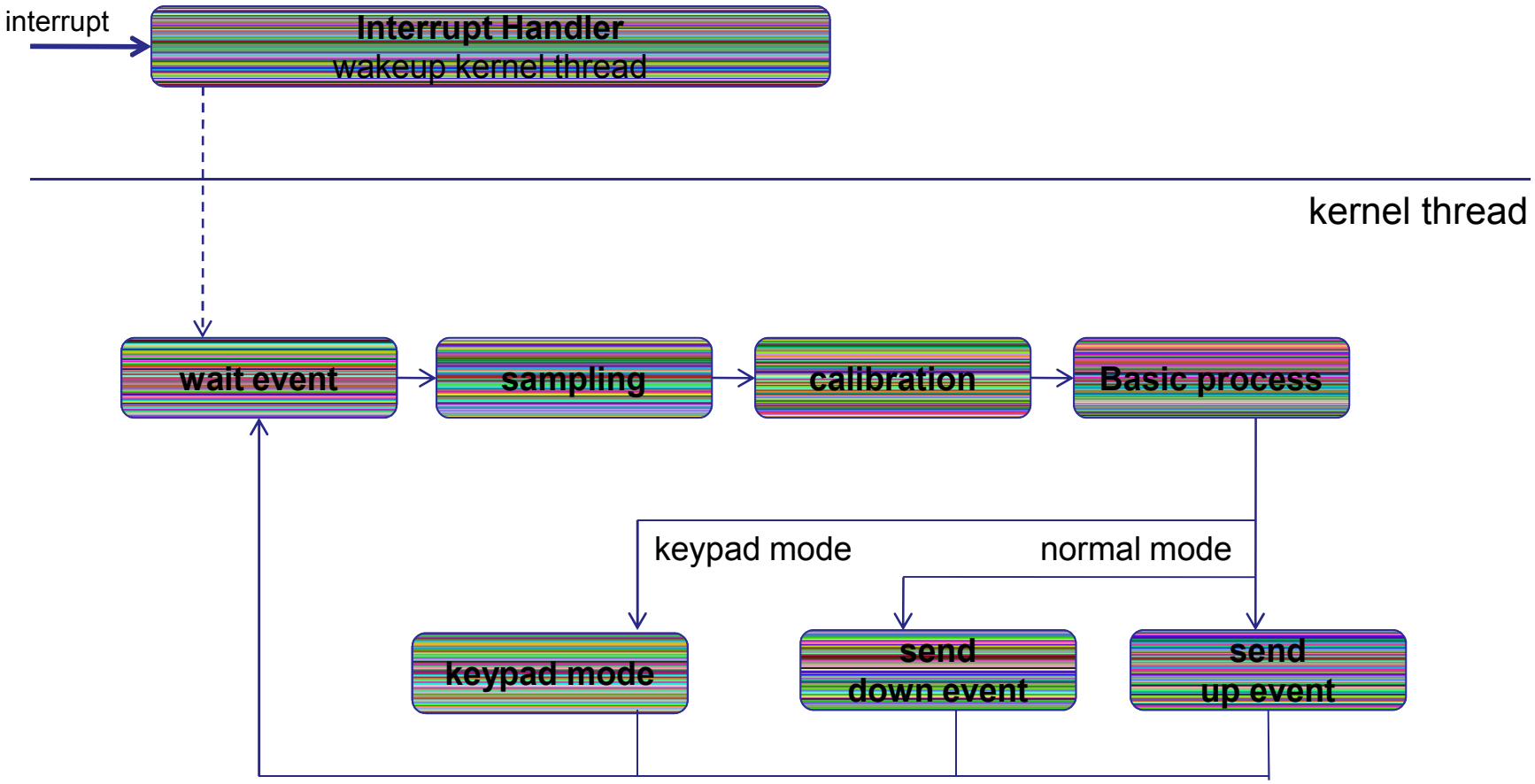
category	name	type	description
	TPD_DELAY	int	in jiffies, next timeout value for tasklet. It controls event rate; faster event rate with smaller TPD_DELAY
Pressure Related	TPD_PRESSURE_MAX	int	Defines the maximum pressure that can be generated by touch panel.
	TPD_PRESSURE_MIN	Int	Defines the minimum pressure that can be generated by touch panel.
	TPD_PRESSURE_NICE	Int	Defines the “nice” pressure of event. If event has larger pressure value than TPD_PRESSURE_NICE, it will be queued and judged whether it is a valid event by following event.

C-type Touch Panel Customization

category	name	type	description
	TPD_POWER_SOURCE	int	Define power source of touch panel component. Refer to mt6516_pll.h for detail power source list.
	TPD_I2C_NUMBER	int	I2c controller number touch panel is on. It should be 0, 1, or 2.

Touch Panel Driver Flow – Event Handling

C-Type touch panel



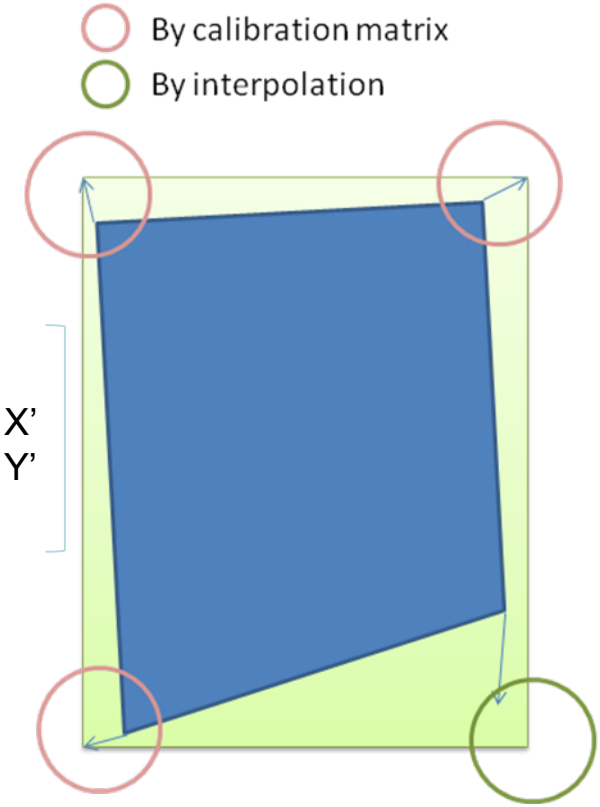
Calibration

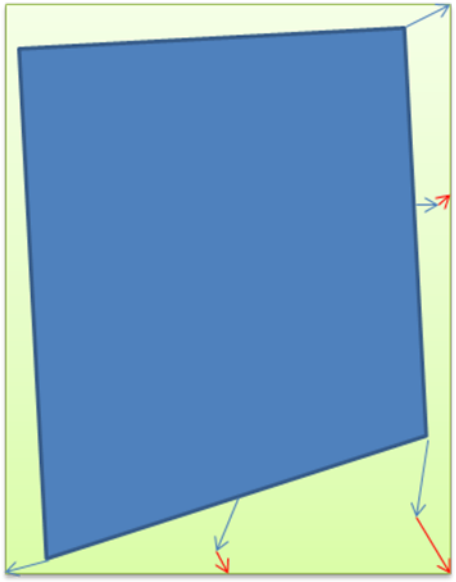
Calibration:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \frac{\begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}}{4096}$$

$$\begin{bmatrix} X'' \\ Y'' \end{bmatrix} = \begin{bmatrix} 1 & \frac{(\mathbf{V}x)X'}{\mathbf{TPD\_RES\_x*TPD\_RES\_y}} \\ \frac{(\mathbf{V}y)Y'}{\mathbf{TPD\_RES\_x*TPD\_RES\_y}} & 1 \end{bmatrix} \begin{bmatrix} X' \\ Y' \end{bmatrix}$$

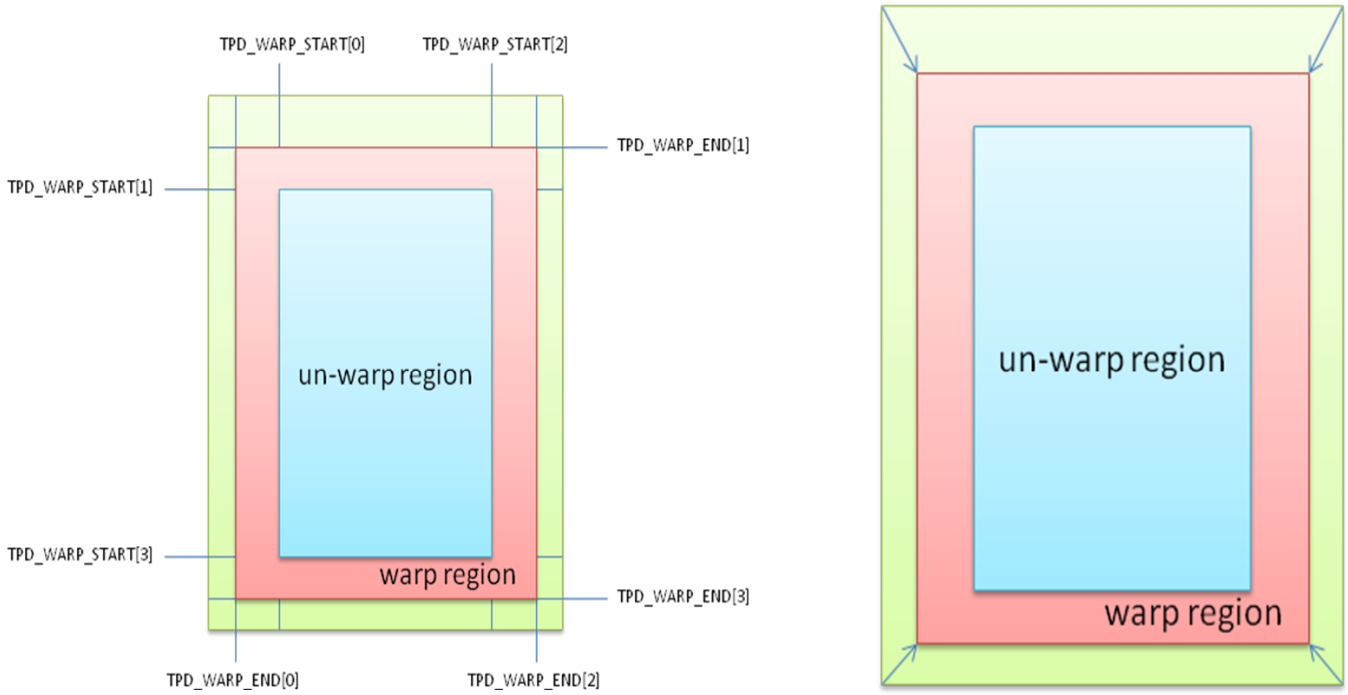
TPD_CALIBRATION_MATRIX:
{ A, B, C, D, E, F, Vx, Vy }





Calibration

Warping:



Points around the edge (green + read region) will be warped into warp region (red region).

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- ***Display***

Connecting
- WIFI
- MT6620 Customization

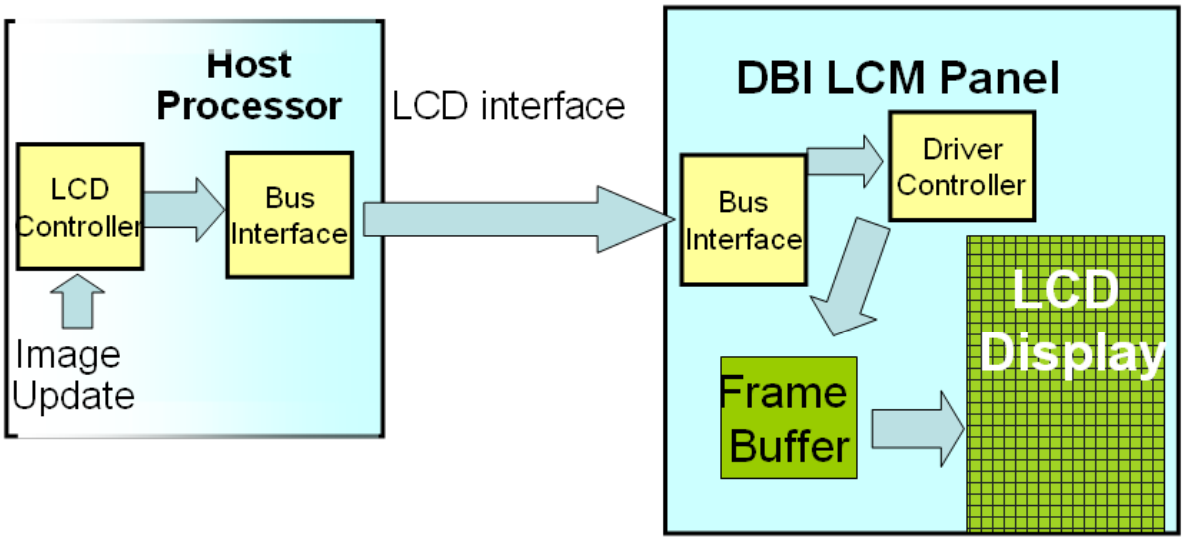
Sensor System
- Sensor Hal
- Sensor Driver Customization

Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

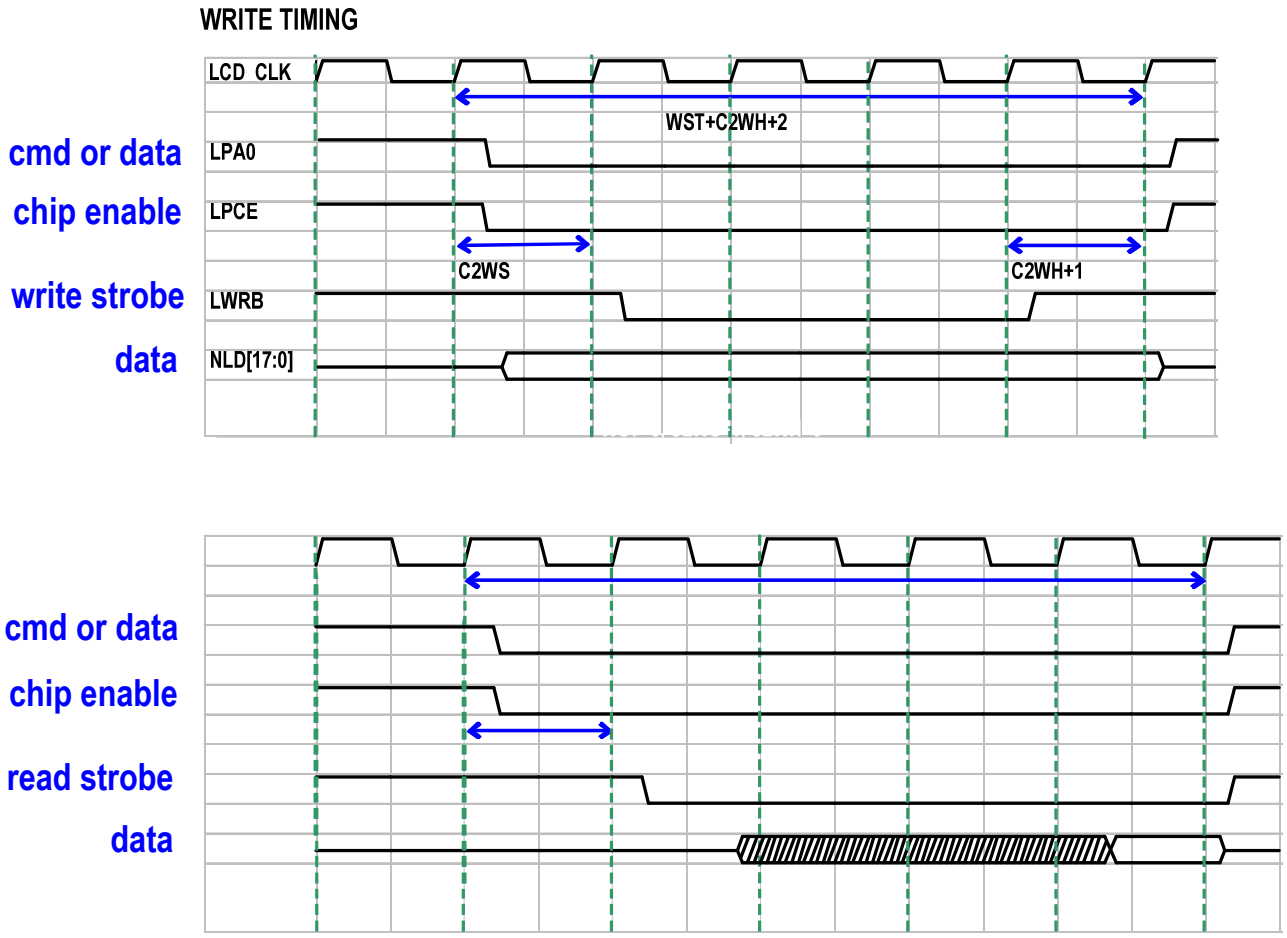
DBI (Display Bus Interface) LCM (1/3)



LCM equips with its own RAM

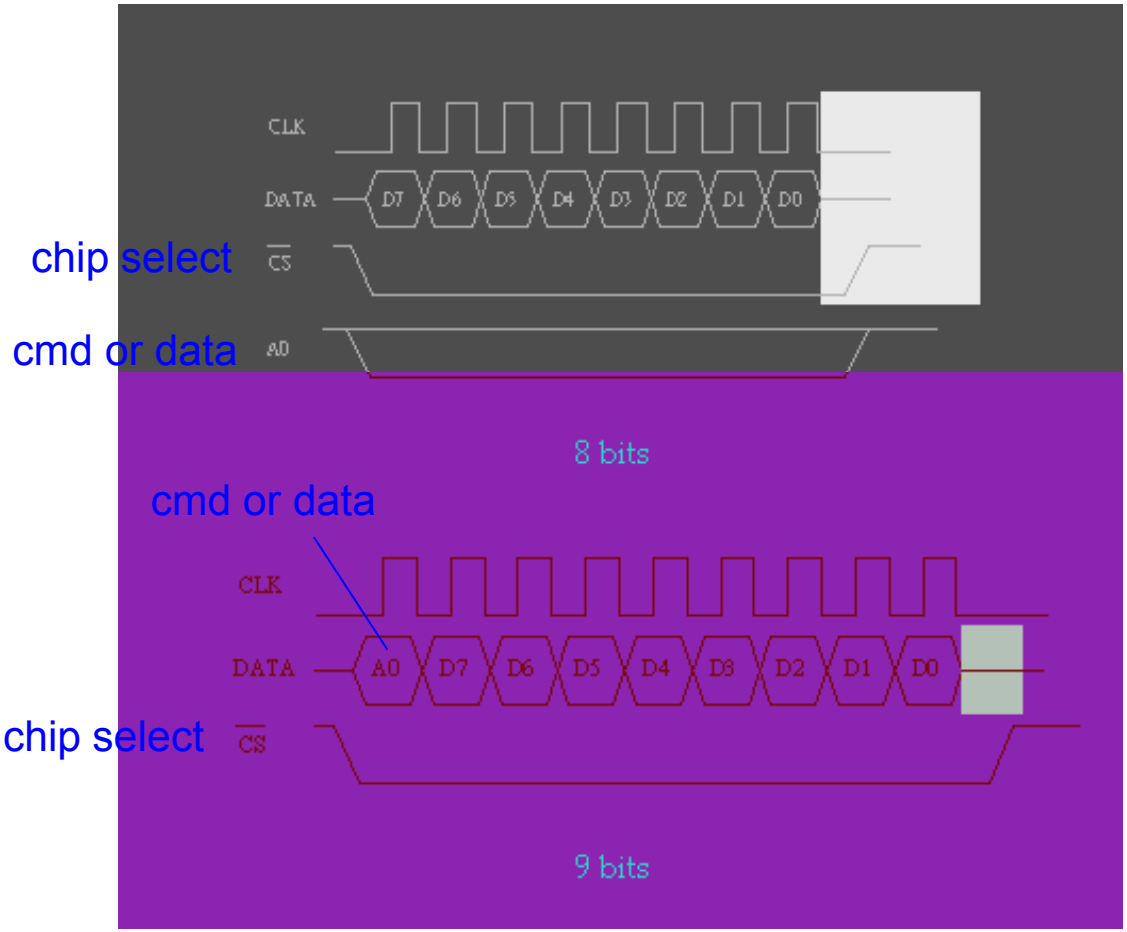
DBI (Display Bus Interface) LCM (2/3)

- **DBI interface timing (parallel I/F)**

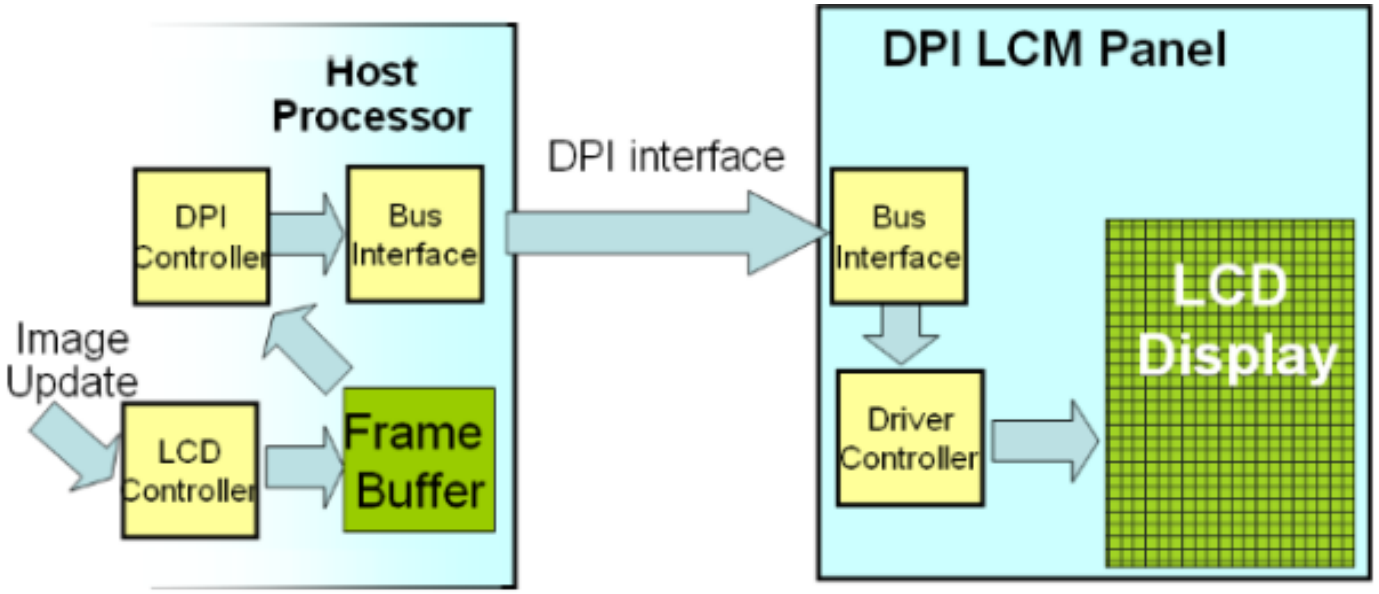


DBI (Display Bus Interface) LCM (3/3)

- DBI interface timing (8-bit / 9-bit serial I/F)



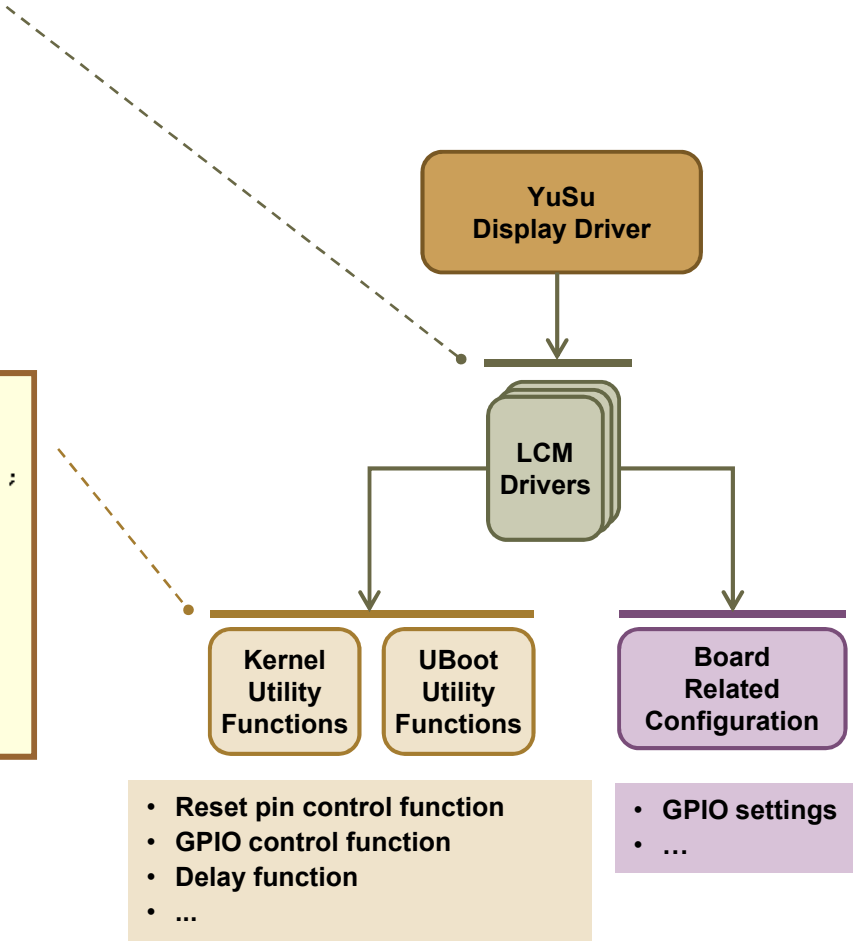
DPI (Display Pixel Interface) LCM



LCM Driver Interface

```
typedef struct
{
    void (*set_util_funcs)(const LCM_UTIL_FUNCS *util);
    void (*get_params)(LCM_PARAMS *params);

    void (*init)(void);
    void (*suspend)(void);
    void (*resume)(void);

    void (*update)(unsigned int x, unsigned int y,
                   unsigned int width, unsigned int height);
} LCM_DRIVER;
```

```
typedef struct
{
    void (*set_reset_pin)(unsigned int value);
    int  (*set_gpio_out)(unsigned int gpio, unsigned int value);

    void (*udelay)(unsigned int us);
    void (*mdelay)(unsigned int ms);

    void (*send_cmd)(unsigned int cmd);
    void (*send_data)(unsigned int data);
    unsigned int (*read_cmd)(void);
    unsigned int (*read_data)(void);
} LCM_UTIL_FUNCS;
```

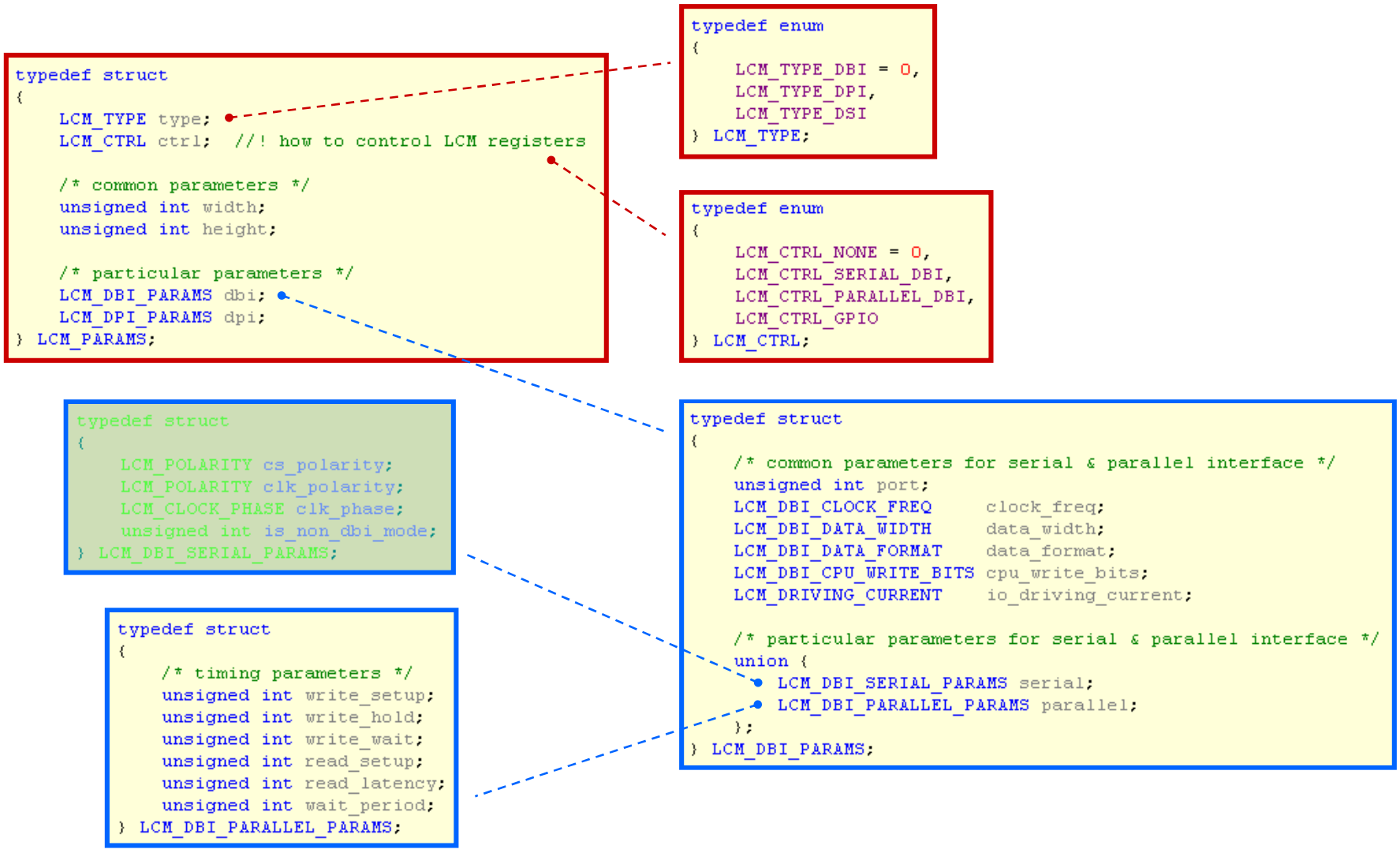


LCM Driver Interface

```
typedef struct
{
    void (*set_util_funcs)(const LCM_UTIL_FUNCS *util);
    void (*get_params)(LCM_PARAMS *params);

    void (*init)(void);
    void (*suspend)(void);
    void (*resume)(void);

    void (*update)(unsigned int x, unsigned int y,
                   unsigned int width, unsigned int height);
} LCM_DRIVER;
```

Name	Description
set_util_funcs	Set LCM utility function interface to LCM driver
get_params	Return LCM parameters for display driver to initialize related HW controllers
init	Initialize the LCM
suspend	Suspend the LCM
resume	Resume the LCM
update	Send the block update commands to LCM

LCM Parameters (Common and DBI)



LCM Parameters (DPI)

```
typedef struct
{
    /*
        Pixel Clock Frequency = 26MHz * mipi_pll_clk_div1
                                      / (mipi_pll_clk_ref + 1)
                                      / (2 * mipi_pll_clk_div2)
                                      / dpi_clk_div
    */
    unsigned int mipi_pll_clk_ref;   // 0..1
    unsigned int mipi_pll_clk_div1;  // 0..63
    unsigned int mipi_pll_clk_div2;  // 0..15
    unsigned int dpi_clk_div;        // 2..32

    unsigned int dpi_clk_duty;       // (dpi_clk_div - 1) .. 31
    /* polarity parameters */
    LCM_POLARITY clk_pol;
    LCM_POLARITY de_pol;
    LCM_POLARITY vsync_pol;
    LCM_POLARITY hsync_pol;
    /* timing parameters */
    unsigned int hsync_pulse_width;
    unsigned int hsync_back_porch;
    unsigned int hsync_front_porch;
    unsigned int vsync_pulse_width;
    unsigned int vsync_back_porch;
    unsigned int vsync_front_porch;
    /* output format parameters */
    LCM_DPI_FORMAT format;
    LCM_COLOR_ORDER rgb_order;
    unsigned int is_serial_output;
    /* intermediate buffers parameters */
    unsigned int intermediat_buffer_num; // 2..3

    /* iopad parameters */
    LCM_DRIVING_CURRENT io_driving_current;

} LCM_DPI_PARAMS;
```

pixel clock frequency

polarity

blanking timing

output color format

misc.

LCM Utility Function Interface

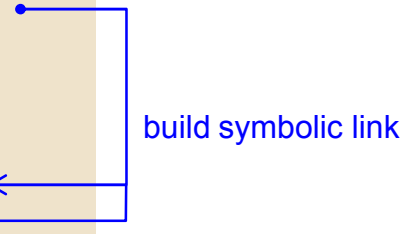
```
typedef struct
{
    void (*set_reset_pin)(unsigned int value);
    int  (*set_gpio_out)(unsigned int gpio, unsigned int value);

    void (*udelay)(unsigned int us);
    void (*mdelay)(unsigned int ms);

    void (*send_cmd)(unsigned int cmd);
    void (*send_data)(unsigned int data);
    unsigned int (*read_cmd)(void);
    unsigned int (*read_data)(void);
} LCM_UTIL_FUNCS;
```

Name	Description
set_reset_pin	Output value to the LCM reset pin
set_gpio_out	Output value to the specified GPIO pin
udelay	Delay several microseconds
mdelay	Delay several milliseconds
send_cmd	Write command to the LCM
send_data	Write data to the LCM
read_cmd	Read command from the LCM
read_data	Read data from the LCM

LCM Customer Folder

- Put all LCM drivers in the custom common kernel folder
- Select LCM by modifying project make file
  - e.g. ./mediatek/config/mt6573_evb/ProjectConfig.mk
    CUSTOM_KERNEL_LCM = bm8587

```
\mediatek\custom\common\kernel\lcm\bm8578\lcm_drv.c
\mediatek\custom\common\kernel\lg4571\lcm_drv.c
\mediatek\custom\common\kernel\s6d0170\lcm_drv.c
...

\bootable\bootloader\uboot\board\mt6516\lcm_drv.c
\kernel\drivers\mediatek\lcm\lcm_drv.c
```



Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- ***WIFI***
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

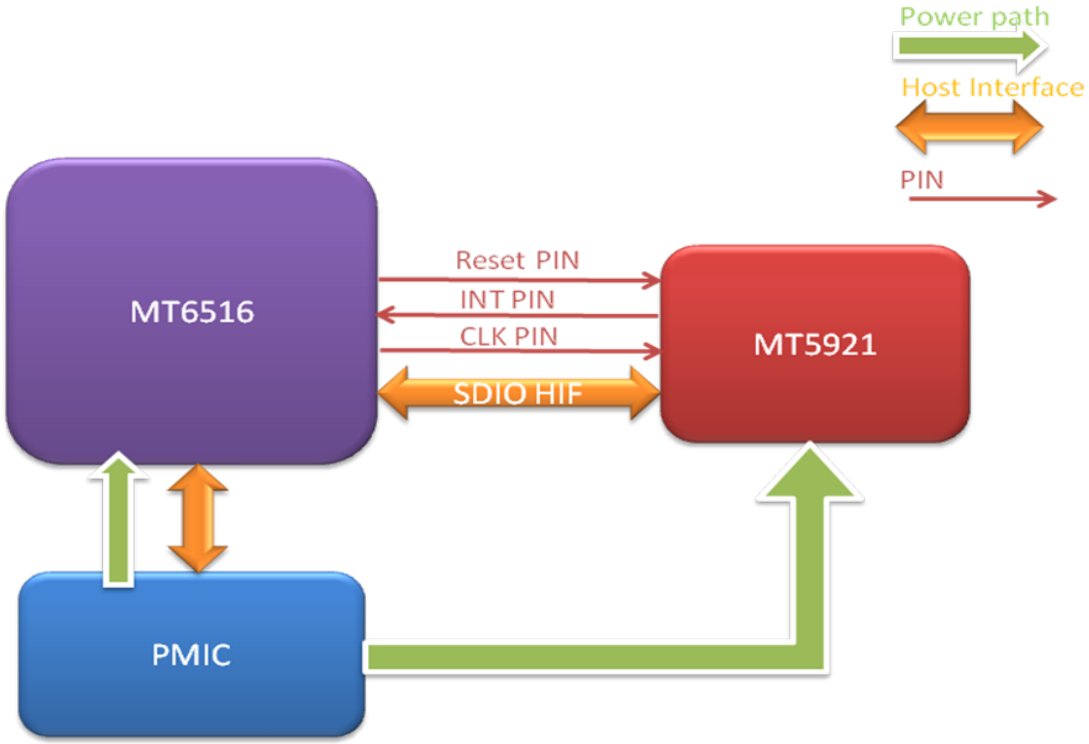
Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

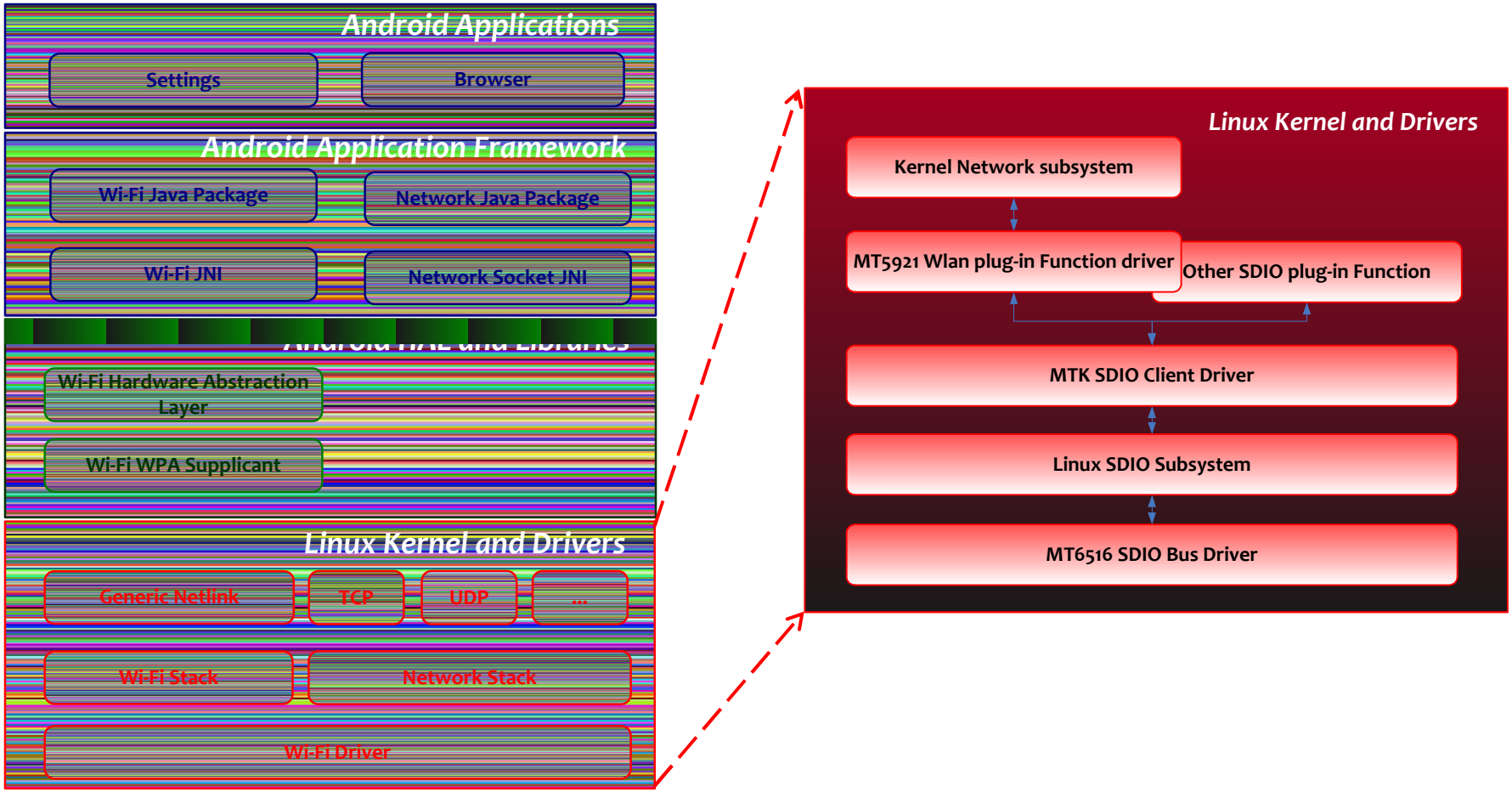
Wi-Fi customization(1/5)

- Wi-Fi Hardware Environment



Wi-Fi customization(2/5)

- Wi-Fi Software Architecture



Wi-Fi customization(3/5)

- Wi-Fi GPIO Setting
  - File: mtk\src\custom\[Project]\kernel\dct\dct\cust_gpio_usage.h
  - 32K Clock source PIN:
    - GPIO_WIFI_CLK_PIN
  - H/W Reset PIN:
    - GPIO_WIFI_RST_PIN
  - External interrupt:
    - GPIO_WIFI_EINT_PIN
- The PIN settings should be defined with DCT tool.

Wi-Fi customization(4/5)

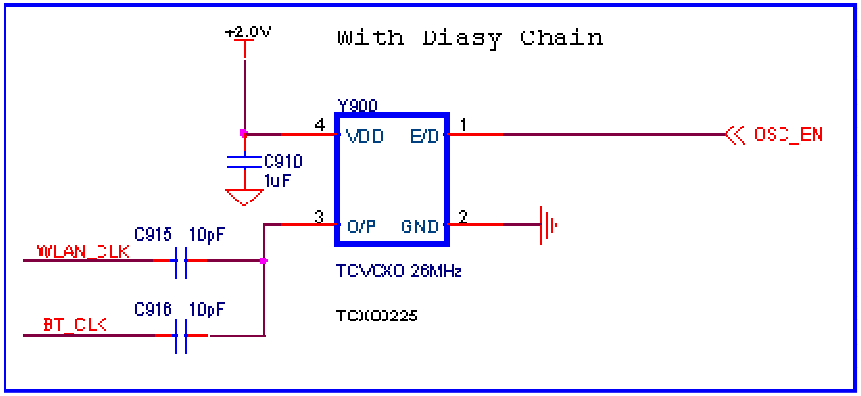
- Wi-Fi Power on/off procedure
  - **Power On**
    - File: mtk\src\custom\[Project]\kernel\core\src\board.c
    - Function : ***mt5921_power_on()***
  - **Power Off**
    - File: mtk\src\custom\[Project]\kernel\core\src\board.c :
    - Function : ***mt5921_power_off()***
- Single/Dual Antenna setting
  - File: mtk\src\custom\[Project]\cgen\inc\wifi_custom.h
  - #define WIFI_CUSTOM_SINGLE_ANT 0
    - This definition means that current project use dual antenna
  - #define WIFI_CUSTOM_SINGLE_ANT 1
    - This definition means that BT and Wi-Fi share the same antenna

Wi-Fi customization(5/5)

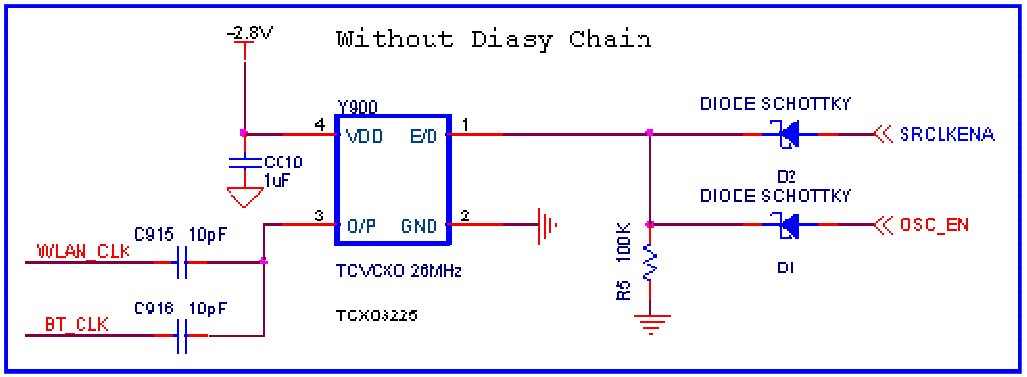
- **LED Indication setting**
  - File: mtk\src\custom\[Project]\cgen\inc\wifi_custom.h
  - #define WIFI_CUSTOM_LED_BLINK_MODE *X*
    - This definition is used to specify LED blinking mode. *X* value can be :
      - 0 : LED does not blink
      - 1 : LED blinks when TX
      - 2 : LED blinks when RX
      - 3 : LED blinks when TX or RX
    - Note : if there is no LED connected to MT5921 in H/W, please define this macro to 0.
  - #define WIFI_CUSTOM_LED_BLINK_ON_TIME 80
    - This definition is used to specify LED blinking on time, unit is ms, range 0~1020.
    - Default setting is 80.
  - #define WIFI_CUSTOM_LED_BLINK_OFF_TIME 24
    - This definition is used to specify LED blinking off time, unit is ms, range 0~1020
    - Default setting is 24.

Daisy chain setting

- **Daisy Chain setting**
  - File: mtk\src\custom\[Project]\cgen\inc\wifi_custom.h
  - #define COMODULE_DAISY_CHAIN 1
    - This definition is used to specify that the daisy chain function is used.
  - #define COMODULE_DAISY_CHAIN 0
    - This definition is used to specify that the daisy chain function is not used.
  - Note : you can decide whether daisy chain function is used from schematic :



with Daisy chain



without Daisy Chain

Source code Structure & File Description

File	
mtk\src\custom\[Project]\kernel\dct\dct\	
cust_gpio_usage.h	
mtk\src\custom\[Project]\kernel\dct\dct\	
cust_eint.h	
mtk\src\custom\[project]\kernel\core\src\	
board.c	
mtk\src\custom\[Project]\kernel\wifi\mt592x\	
wifi_custom.h	
kernel\drivers\net\wireless\mt592x\	
Platform.c	
kernel\drivers\net\wireless\mt592x\	
Hif.c	
kernel\drivers\net\wireless\mt592x\	
Wlan.ko	

Outline

Overview
- Chip
- Android
- Build Command

Boot up
- Bootloader
- Android Booting

Lights System

Battery Manager
- JNI
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display
- UART

Connecting
- WIFI
- ***MT6620 Customization***

Sensor System
- Sensor Hal
- Sensor Driver Customization

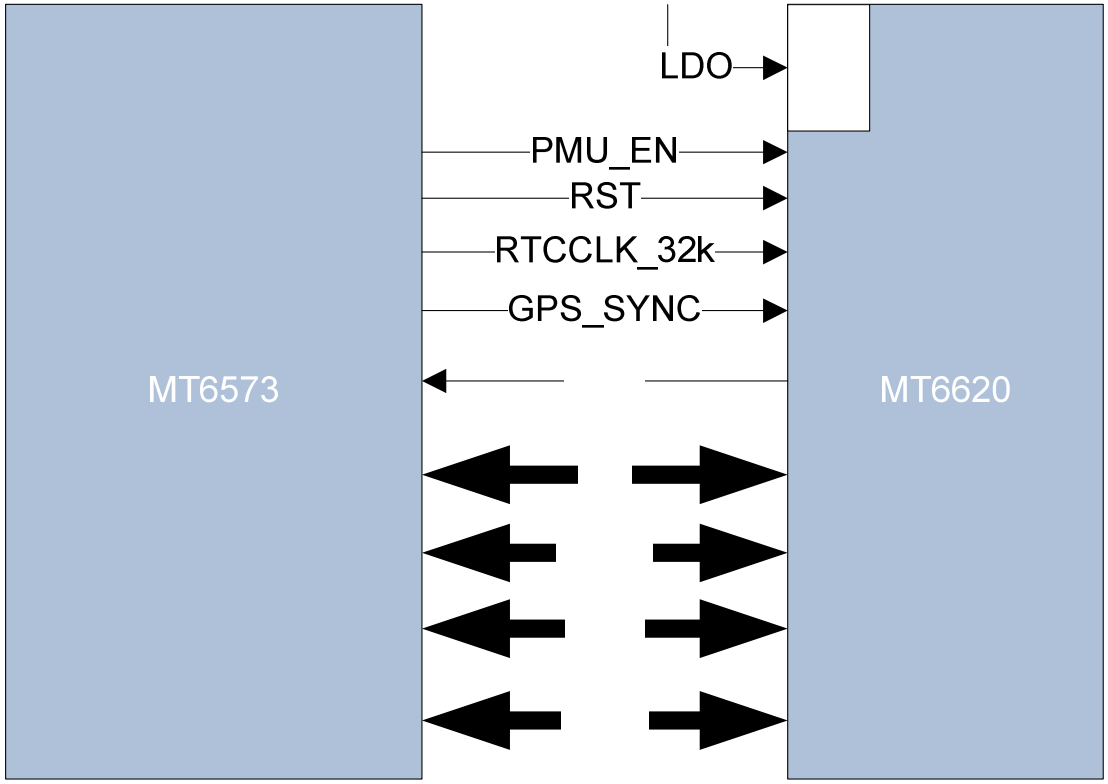
Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

MT6620 Hardware Environment

- MT6620= BT+WIFI+GPS+FM



MT6620 SW Architecture

Application for New Features

Application Framework for New Features

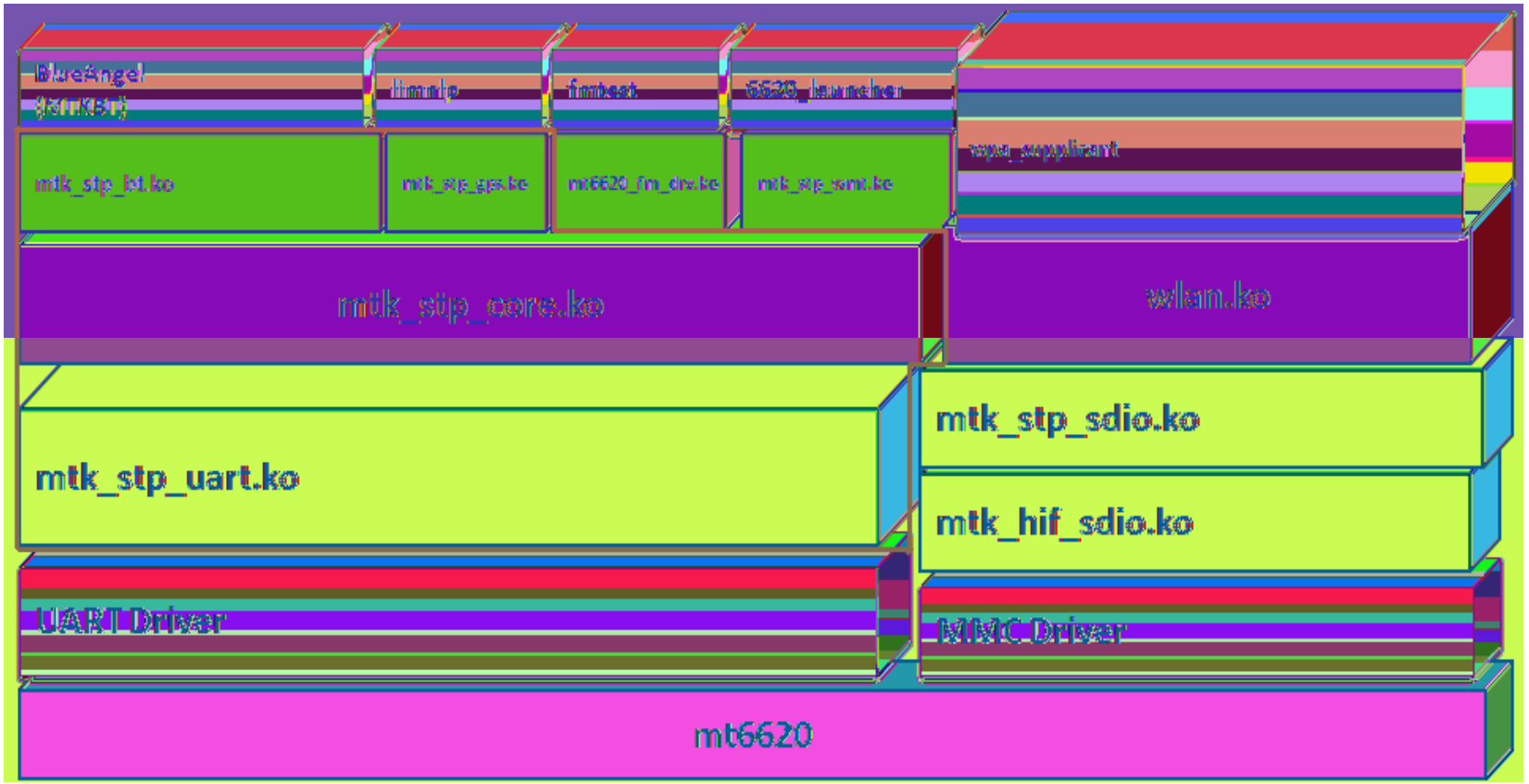
Application Specific

Native Applications

Kernel Drivers

Common for MT6620

Detail Architecture



GPIO pins customization

- LDO control pin
  - GPIO_COMBO_6620_LDO_EN_PIN
- Uart
  - GPIO_UART_UTXD3_PIN
  - GPIO_UART_URXD3_PIN
- PCM: (for BT)
  - GPIO_PCM_DAICLK_PIN
  - GPIO_PCM_DAIPCMOUT_PIN
  - GPIO_PCM_DAIPCMIN_PIN
  - GPIO_PCM_DAISYNC_PIN
- External interrupt
  - GPIO_COMBO_BGF_EINT_PIN
  - GPIO_WIFI_EINT_PIN
- RTC clock source
  - GPIO_COMBO_RTCCLK_PIN
- Power enable pin and reset pin
  - GPIO_COMBO_PMU_EN_PIN
  - GPIO_COMBO_RST_PIN
- GPS SYNC PIN
  - GPIO_GPS_SYNC_PIN

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- ***Sensor Hal***
- Sensor Driver Customization

Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

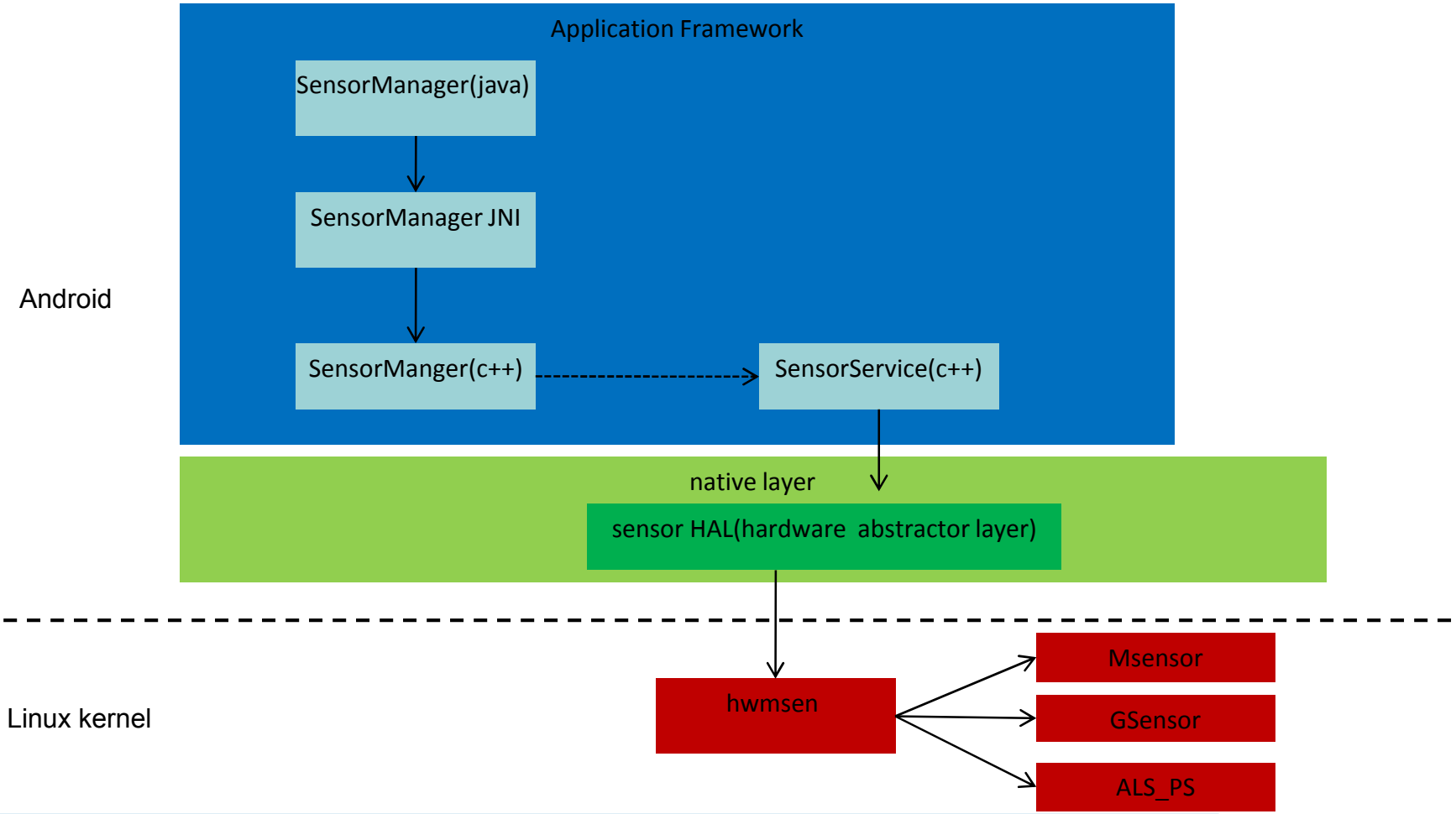
Android sensor support types

•Now Android support 8 types sensors.

Sensor types	Service define	Driver define
Accelerometer	TYPE_ACCELEROMETER	SENSOR_TYPE_ACCELEROMETER
Magnetic	TYPE_MAGNETIC_FIELD	SENSOR_TYPE_MAGNETIC_FIELD
Orientation	TYPE_ORIENTATION	SENSOR_TYPE_ORIENTATION
Gyroscope	TYPE_GYROSCOPE	SENSOR_TYPE_GYROSCOPE
Light	TYPE_LIGHT	SENSOR_TYPE_LIGHT
Pressure	TYPE_PRESSURE	SENSOR_TYPE_PRESSURE
Temperature	TYPE_TEMPERATURE	SENSOR_TYPE_TEMPERATURE
Proximity	TYPE_PROXIMITY	SENSOR_TYPE_PROXIMITY

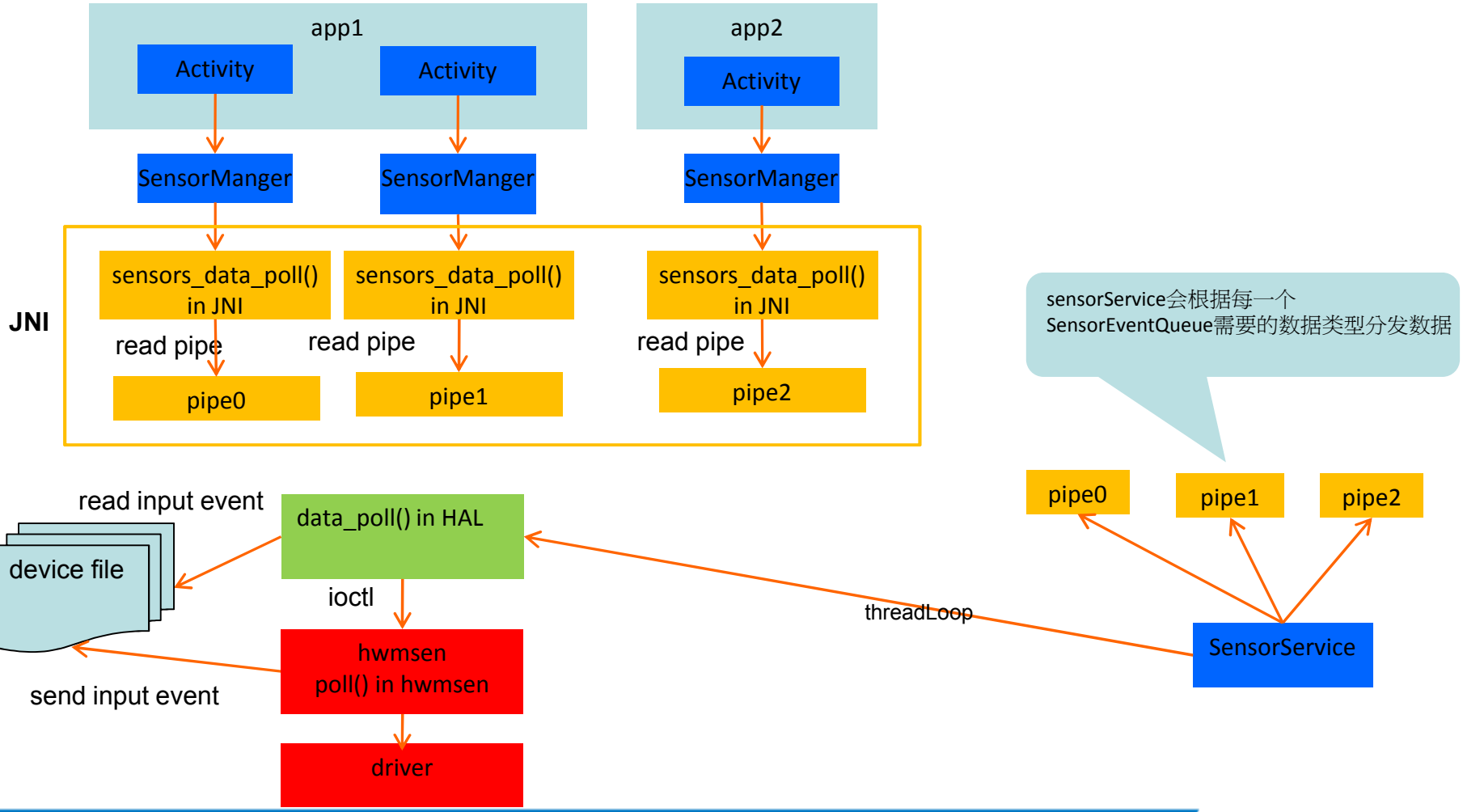
Sensor system Architecture

- sensor Architecture (android 2.3)



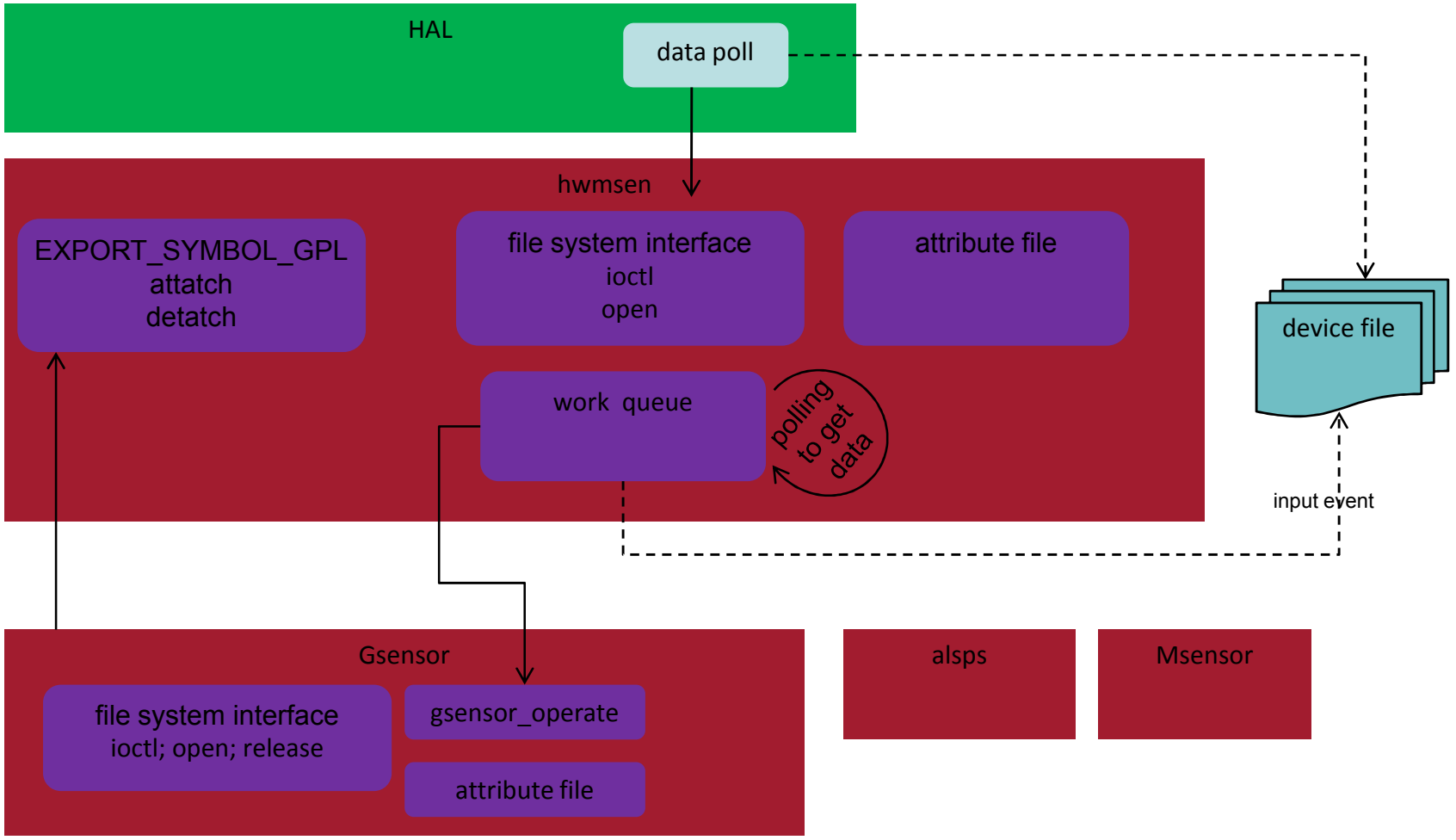
Sensor Manager

- JNI(android2.3)



Hwmsen driver(1/5)

- architecture



Sensor HAL Customization

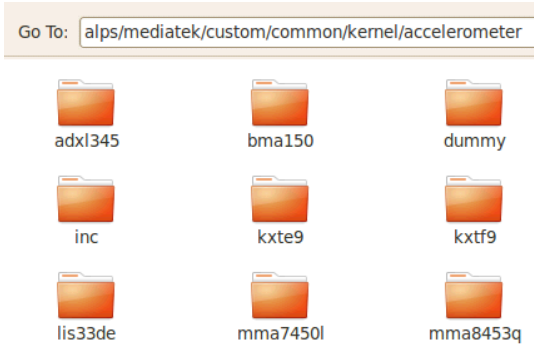
- Makefile Customization
  - Alps/mediatek/config/$(project)/ProjectConfig.mk file set the sensors' configure

```
# Android sensor device
MTK_SENSOR_SUPPORT = yes

CUSTOM_KERNEL_MAGNETOMETER = ami304

CUSTOM_KERNEL_ACCELEROMETER = adxl345

CUSTOM_KERNEL_ALSPS = cm3623

CUSTOM_HAL_SENSORS = sensor

CUSTOM_HAL_MSENSORLIB = ami304
```

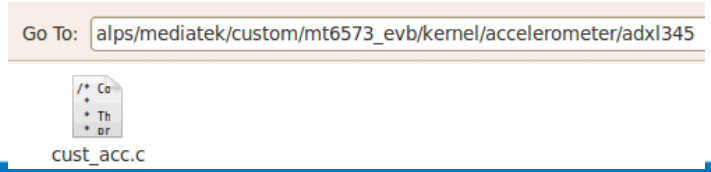
  - If you want to support sensor in your project, please always set
    MTK_SENSOR_SUPPORT = yes
    CUSTOM_HAL_SENSORS = sensor

Sensor HAL Customization

- G sensor driver Customization
  - If project use g sensor adxl345, please set
    - CUSTOM_KERNEL_ACCELEROMETER = adxl345
  - If have no g sensor, set as follow
    - CUSTOM_KERNEL_ACCELEROMETER =
  - G sensor driver is location at
    - G sensor driver

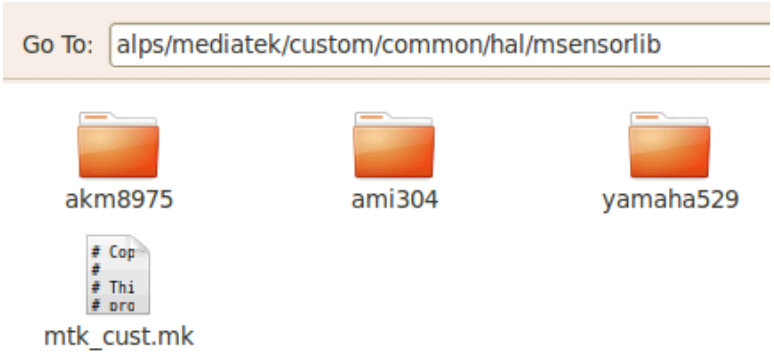


    - Customization file



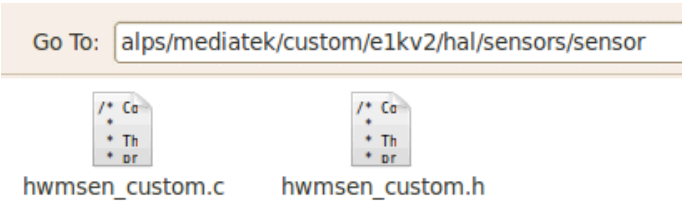
Sensor HAL Customization

- M sensor driver Customization
  - If project use m sensor ami304, please set
    - CUSTOM_KERNEL_MAGNETOMETER = ami304
    - CUSTOM_HAL_MSENSORLIB = ami304
  - If have no g sensor, set as follow
    - CUSTOM_KERNEL_MAGNETOMETER =
    - CUSTOM_HAL_MSENSORLIB =
  - msensor deamoon source code



Sensor HAL Customization

- Sensor Hal Customization
  - Alps/mediatek/custom/$(project)/hal/sensors/sensor folder have project customization configure file



  - Customization the detail information in hwmsens_custom.c, for example

```
struct sensor_t sSensorList[MAX_NUM_SENSORS] =
{
  {
    .name       = "YAMAHA Orientation sensor",
    .vendor     = "Yamaha",
    .version    = 1,
    .handle     = ID_ORIENTATION,
    .type       = SENSOR_TYPE_ORIENTATION,
    .maxRange   = 360.0f,
    .resolution = 1.0f,
    .power      = 0.25f,
    .reserved   = {}
  },
```

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- ***Sensor Driver Customization***

Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

G-Sensor Customization (1/3)

- Change file list

File Name	Location
cust_acc.h	alps\mediatek\custom\common\kernel\accelerometer\inc
cust_acc.c	alps\mediatek\custom\${BOARD}\kernel\accelerometer\${MODULE}
(G sensor driver)	alps\mediatek\custom\common\kernel\accelerometer\${sensor_name}

- Customization item
  - Overview

```
struct acc_hw {
    int i2c_num;
    int direction;
    int power_id;
    int power_vol;
    int firlen;
};
```

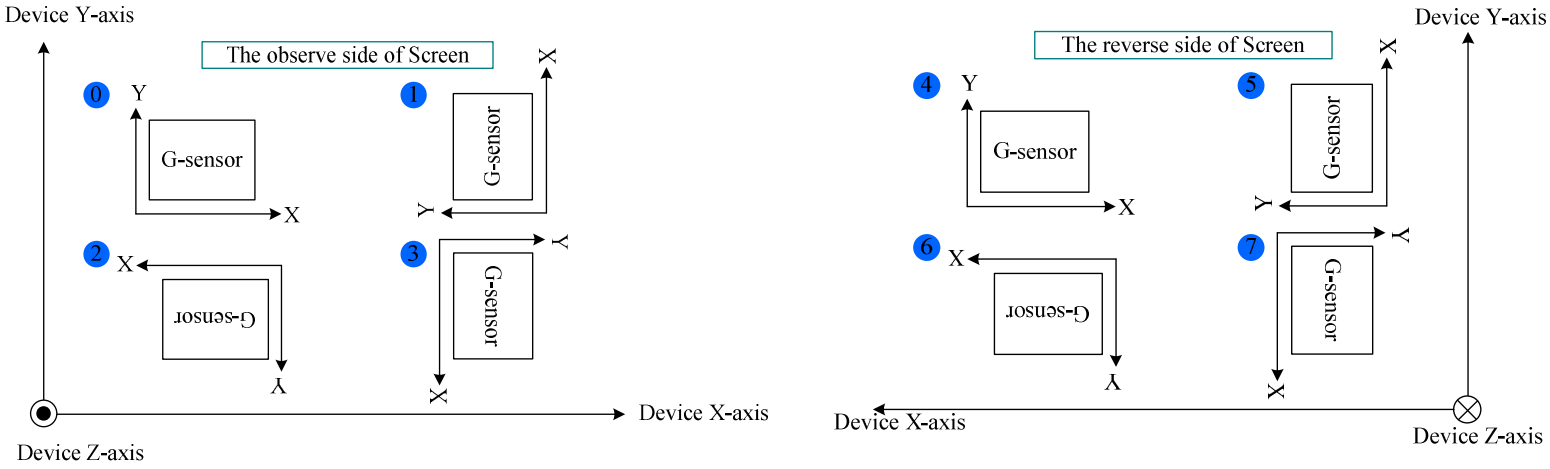
  - i2c_num
    - Customer can define the I2C number used by sensor
    - The value could be defined as 0 ~ 2
  - firlen
    - Customer can define the filter length of SW low pass filter.
    - The value could be defined as 0 ~ 32. 0 will disable the functioality.

G-Sensor Customization (2/3)

– direction

- Customer can define the device direction of g-sensor in device.
- The value could be defined as 0 ~ 7

Value	Description
0	{x, y, z} => { x, y, z}
1	{x, y, z} => {-y, x, z}
2	{x, y, z} => {-x, -y, z}
3	{x, y, z} => { y, -x, z}
4	{x, y, z} => {-x, y, -z}
5	{x, y, z} => { y, x, -z}
6	{x, y, z} => { x, -y, -z}
7	{x, y, z} => {-y, -x, -z}



G-Sensor Customization (3/3)

- power_id / power_vol
  - Customer could define power source of device according to layout
  - Please refer to the following file for power id and voltage
    - alps\mediatek\platform\mt6573\core\include\mach\mt6573_pll.h
  - If the power source can't be shutdown, please set the power_id as MT65XX_POWER_NONE

```
#include <cust_acc.h>
#include <mach/mt6573_pll.h>
/*----------------------------------------------------------------------------*/
int cust_acc_power(struct acc_hw *hw, unsigned int on, char* devname)
{
    if (hw->power_id == MT65XX_POWER_NONE)
        return 0;
    if (on)
        return hwPowerOn(hw->power_id, hw->power_vol, devname);
    else
        return hwPowerDown(hw->power_id, devname);
}
```

M-Sensor Customization (1/3)

- Change file list

File Name	Location
cust_mag.h	alps\mediatek\custom\common\kernel\magnetometer\inc
cust_mag.c	alps\mediatek\custom\${BOARD}\kernel\magnetometer\${MODULE}

- Customization item
  - Overview

```
struct mag_hw {
    int i2c_num;
    int direction;
    int power_id;
    int power_vol;
};
```

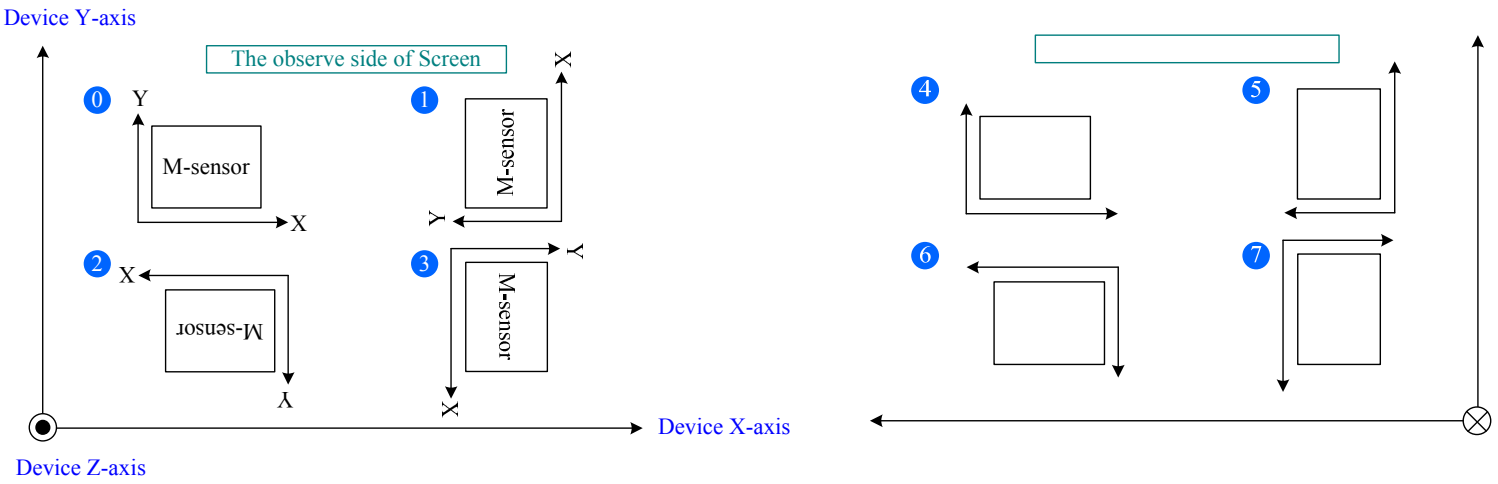
  - i2c_num
    - Customer can define the I2C number used by sensor
    - The value could be defined as 0 ~ 2

M-Sensor Customization (2/3)

– direction

- Customer can define the device direction of sensor in device.
- The value could be defined as 0 ~ 7

Value	Description
0	{x, y, z} => { x, y, z}
1	{x, y, z} => {-y, x, z}
2	{x, y, z} => {-x, -y, z}
3	{x, y, z} => { y, -x, z}
4	{x, y, z} => {-x, y, -z}
5	{x, y, z} => { y, x, -z}
6	{x, y, z} => { x, -y, -z}
7	{x, y, z} => {-y, -x, -z}



M-Sensor Customization (3/3)

- power_id / power_vol
  - Customer could define power source of device according to layout
  - Please refer to the following file for power id and voltage
    - arch\arm\mach\include\mach-mt6516\include\mach\mt6516_pll.h
  - If the power source can't be shutdown, please set the power_id as MT6516_POWER_NONE

ALS/PS Customization (1/4)

- Change file list

File Name	Location
cust_alsps.h	alps\mediatek\custom\common\kernel\alsps\inc
cust_alsps.c	alps\mediatek\custom\${BOARD}\kernel\alsps\${MODULE}

- Customization item
  - Overview

```
#define C_CUST_ALS_LEVEL    16
#define C_CUST_I2C_ADDR_NUM 4

struct alsps_hw {
    int i2c_num;
    int power_id;
    int power_vol;
    unsigned char   i2c_addr[C_CUST_I2C_ADDR_NUM];
    unsigned int    als_level[C_CUST_ALS_LEVEL-1];
    unsigned int    als_value[C_CUST_ALS_LEVEL];
    unsigned int    ps_threshold;
};
```

  - i2c_num
    - Customer can define the I2C number used by sensor
    - The value could be defined as 0 ~ 2

ALS/PS Customization (2/4)

- power_id / power_vol
  - Customer could define power source of device according to layout
  - Please refer to the following file for power id and voltage
    - arch\arm\mach\include\mach-mt6516\include\mach\mt6516_pll.h
  - If the power source can't be shutdown, please set the power_id as MT6516_POWER_NONE
- i2c_addr
  - This is an array of i2c address used in ALS+PS sensor.
    - Some component (CM3623) owns more than one i2c address
- ps_threshold
  - The threshold is used to judge if object is close or not.
  - If the value reported by proximity sensor is larger than **ps_threshold**, it means the object is close. Otherwise, the object is far away.
  - The actual value range depends on each sensor

ALS/PS Customization (3/4)

- als_level & als_value
  - The two items will remap the raw data to range 0.0 ~ 10240.0.
  - The (C_CUST_ALS_LEVEL-1) values in als_level will divide [0.0 10240.0] into C_CUST_ALS_LEVEL zones. The values in als_value will be reported if the raw data falls into the corresponding zones.
  - The framework will use the remapped value to adjust screen backlight/keypad/button backlight

Driver Level	Driver Value	Framework Level	Framework Value
0	40	0	30
0	40	16	40
0	90	32	50
0	90	50	60
0	160	100	70
0	160	140	80
50	225	180	102
100	320	240	102
1000	640	300	102
2000	1280	600	102
3000	1280	1000	102
6000	2600	2000	180
10000	2600	3000	200
14000	2600	4000	210
18000	10240	8000	230
20000	10240	10000	255

An example of als_value & als_level

ALS/PS Customization (4/4)

- DCT Customization

DCT definition	Description
GPIO_ALS_EINT_PIN	The GPIO pin for ALS EINT (external interrupt)
CUST_EINT_ALS_NUM	The ID of ALS EINT
CUST_EINT_ALS_DEBOUNCE_CN	The debounce count of ALS. It's set as **0x00** for CM3623
CUST_EINT_ALS_POLARITY	The polarity of ALS. It's set as **low level** for CM3623
CUST_EINT_ALS_SENSITIVE	The sensitivity of ALS. It's set as **level sensitive** for CM3623
CUST_EINT_ALS_DEBOUNCE_EN	Enable / disable the debounce. It's set as **disable** for CM3623

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

Audio

Misc.
- GPIO
- I2C
- PMIC
- Headset

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

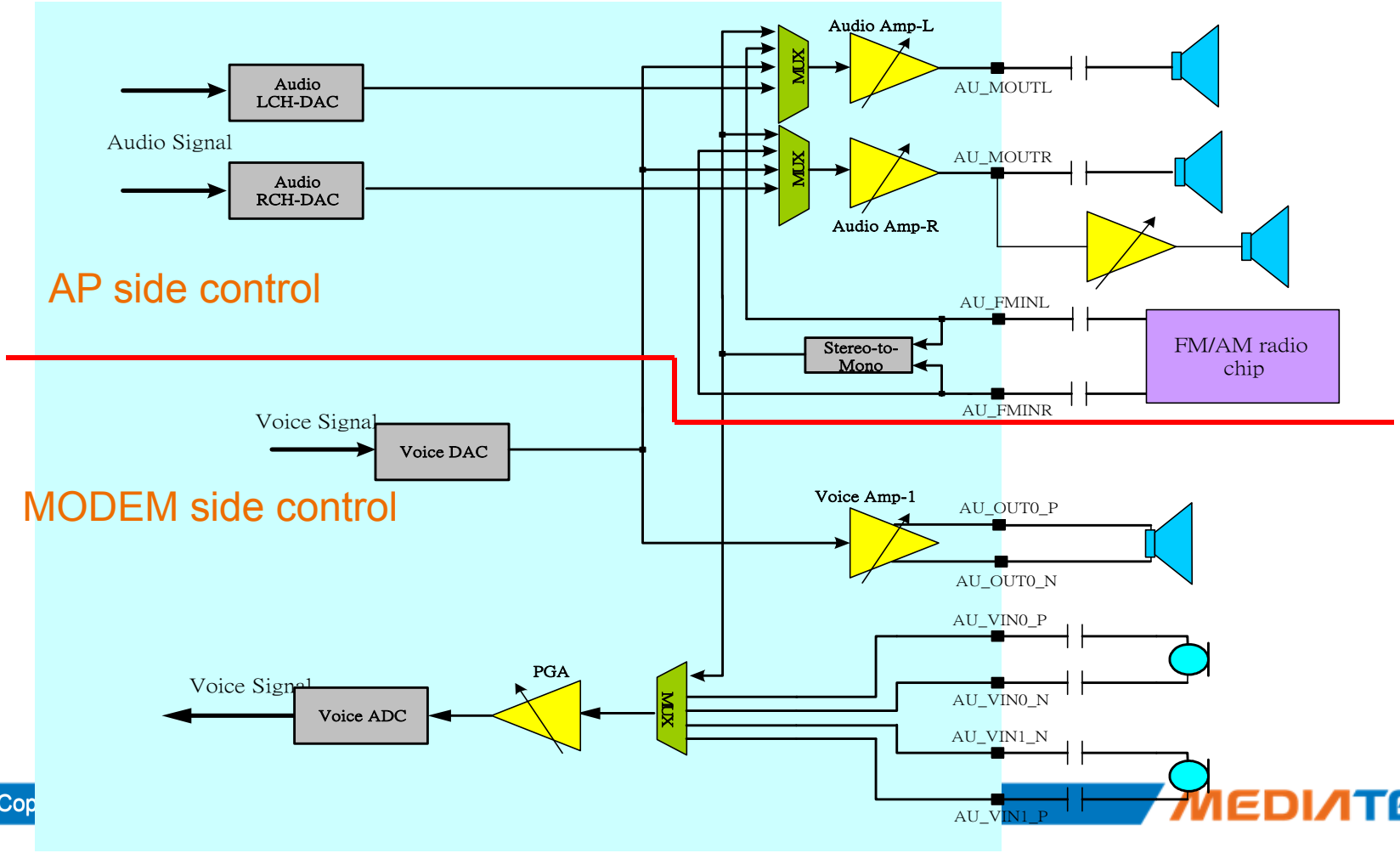
Introduction

- On MT6516, AP side(ARM9) control audio path, MODEM side(ARM7) control speech path. MODEM side works as the slave of AP side.

- AP side send command to MODEM side. All kinds of speech related configuration and control flow are controlled by AP side. Ex: volume setting, speech on/off, mode setting,…

Audio Path

- Hardware Block is the same as feature phone， the only difference is that who the control side is，AP or Modem.



Audio_mode

- Define three mode for audio
  - **Normal**, Idle audio playback or record mode, or just idle mode.
  - **Ringtone**, if there is an in coming call ,it will enter this mode.
  - **In Call**, if it is in a call, no matter MO or MT call ,it will enter this mode.
- PS: Mode can tell us that which path is available ,But it can NOT tell us the exactly path that it uses.
  - Please refer to [Audio_Routing_Policy.doc] for more information.

```
enum audio_mode {
    MODE_INVALID = -2,
    MODE_CURRENT = -1,
    MODE_NORMAL = 0,
    MODE_RINGTONE,
    MODE_IN_CALL,
    NUM_MODES  // not a valid entry, denotes end-of-list
};
```

Force use path

- Force use categories
  - Force use device to which path.

```
// device categories used for setForceUse()
enum forced_config {
    FORCE_NONE,
    FORCE_SPEAKER,
    FORCE_HEADPHONES,
    FORCE_BT_SCO,
    FORCE_BT_A2DP,
    FORCE_WIRED_ACCESSORY,
    FORCE_BT_CAR_DOCK,
    FORCE_BT_DESK_DOCK,
    NUM_FORCE_CONFIG,
    FORCE_DEFAULT = FORCE_NONE
};
```

It is the highest priority path configuration!!!!

- Set force use condition
  - In Communication/Media/Record, when we will use this path.

```
// usages used for setForceUse()
enum force_use {
    FOR_COMMUNICATION,
    FOR_MEDIA,
    FOR_RECORD,
    FOR_DOCK,
    NUM_FORCE_USE
};
```

Speech setting: Tx/Rx FIR

- Currently, Tx/Rx FIR are stored at AP side NVRAM. The data structure is the same as feature phone.
- The way of modifying FIR
  - Use audio tool in **SP META**
  - Modify source code in

File name	Location
audio_custom.h	\mtk\src\custom\[Project Name]\cgen\inc

```
/* 0: Output FIR coefficients for 2G/3G Normal mode */
/* 1: Output FIR coefficients for 2G/3G/VoIP Headset mode */
/* 2: Output FIR coefficients for 2G/3G Handfree mode */
/* 3: Output FIR coefficients for 2G/3G/VoIP BT mode */
/* 4: Output FIR coefficients for VoIP Normal mode */
/* 5: Output FIR coefficients for VoIP Handfree mode */
#define SPEECH_OUTPUT_FIR_COEFF \
   440,  189,  271,  251,   69,        \
   293, -292,   86 , -412, 1013,       \
  -316,-1135,  353,  263,  928,        \
 -2577, 3419,   76, -378,13989,        \
-32767,30692,30692,-32767,13989,       \
  -378,   76, 3419, -2577,  928,       \
   263,  353,-1135,  -316, 1013,       \
  -412,   86, -292,   293,   69,       \
   251,  271,  189,   440,    0,       \
                                       \
```

Speech setting: Speech mode parameter

- There are several output devices supported in speech mode
  - Normal mode
  - Earphone mode
  - Loudspeaker mode
  - BT earphone mdoe
  - BT cordless mode
  - ….
- Different speech enhancement algorithms are applied to different modes respectively. These algorithms are controlled by speech mode parameters
  - For the detail setting of the mode parameters, please refer to Android_SmartPhone_Customization.doc

Volume setting—Analog Gain

- The maxVol should be decided on each platform.
  - By mode: Normal mode, Headset mode and Handfree mode
  - Unit of (-1dB), so the larger the value is , the less the volume will be

Variable name	Definition
GAIN_NOR_TON_VOL	Reserved
GAIN_NOR_KEY_VOL	Reserved
GAIN_NOR_MIC_VOL	IN_CALL BuiltIn Mic gain
GAIN_NOR_FMR_VOL	Idle BuiltIn Mic gain
GAIN_NOR_SID_VOL	IN_CALL EARPIECE Sidetone
GAIN_NOR_SPH_VOL	IN_CALL EARPIECE Volume
GAIN_NOR_MED_VOL	Reserved
GAIN_HED_TON_VOL	Reserved
GAIN_HED_KEY_VOL	Reserved
GAIN_HED_MIC_VOL	IN_CALL headset mic gain
GAIN_HED_FMR_VOL	Reserved
GAIN_HED_SID_VOL	IN_CALL Headset sidetone
GAIN_HED_SPH_VOL	IN_CALL Headset Volume
GAIN_HED_MED_VOL	Reserved
GAIN_HND_TON_VOL	Idle Headset Audio Buf Gain setting
GAIN_HND_KEY_VOL	Reserved
GAIN_HND_MIC_VOL	IN_CALL Handfree MicGain
GAIN_HND_FMR_VOL	Reserved
GAIN_HND_SID_VOL	IN_CALL HandfreeMic sidetone
GAIN_HND_SPH_VOL	IN_CALL HandfreeMic Volume
GAIN_HND_MED_VOL	Idle,Loudspk Audio Buf Gain setting

Audio_Custom.h

Volume setting—Digital Gain

- Digital Gain Setting
  - For example , if we attenuation Notification STREAM for 5dB , user can heard notification sound more small.
    - Provide an array of streamtype in order to attenuate each streamtype.
    - For VOICE_CALL and BLUETOOTH_SCO , we adjust volume for speech parameter , keep value with 0.
    - Other stream can adjust the number , range is 0~200 , it means a stream a have a related range of 100 dB.
  - Adjust for stream_Base can achieve for least volume sound.
    - Value range from 0 ~ 100.

```
// a step means 0.5 dB  ex:20==> attenuation  for 10 dB
static uint32_t Stream_Atten[] =
    {  0, //VOICE_CALL ,no use
       0, //SYSTEM
       0, //RINGONE
       0, //MUSIC
       0, //ALARM
       0, //NOTIFICATION
       0, //BLUETOOTH_SCO , no use
       0, //ENFORCED_AUDIBLE
       0, //DTMF
       0  //TTS
       };

static uint32_t Stream_Base[] =
    {  0, //VOICE_CALL ,no use
       0, //SYSTEM
      30, //RINGONE
      30, //MUSIC
      30, //ALARM
      30, //NOTIFICATION
       0, //BLUETOOTH_SCO , no use
      30, //ENFORCED_AUDIBLE
      30, //DTMF
      30  //TTS
       };
```

File Path :

alps\mtk\src\custom\[Project Name]\hal\audioflinger\audio\audio_custom_exp.h

Volume Setting—Bluetooth

File name	Location
sph_coeff_default.h	alps\mtk\src\custom\[Project Name]\cgen\inc\

```
/* The Bluetooth PCM digital volume */
/* default_bt_pcm_in_vol : uplink, only for enlarge volume,
                           0x100 : 0dB  gain
                           0x200 : 6dB  gain
                           0x300 : 9dB  gain
                           0x400 : 12dB gain
                           0x800 : 18dB gain
                           0xF00 : 24dB gain                */
#define DEFAULT_BT_PCM_IN_VOL        0x100
/* default_bt_pcm_out_vol : downlink gain,
                           0x1000 : 0dB; maximum 0x7FFF   */
#define DEFAULT_BT_PCM_OUT_VOL       0x1000
#endif
```

Too high gain will cause distortion

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

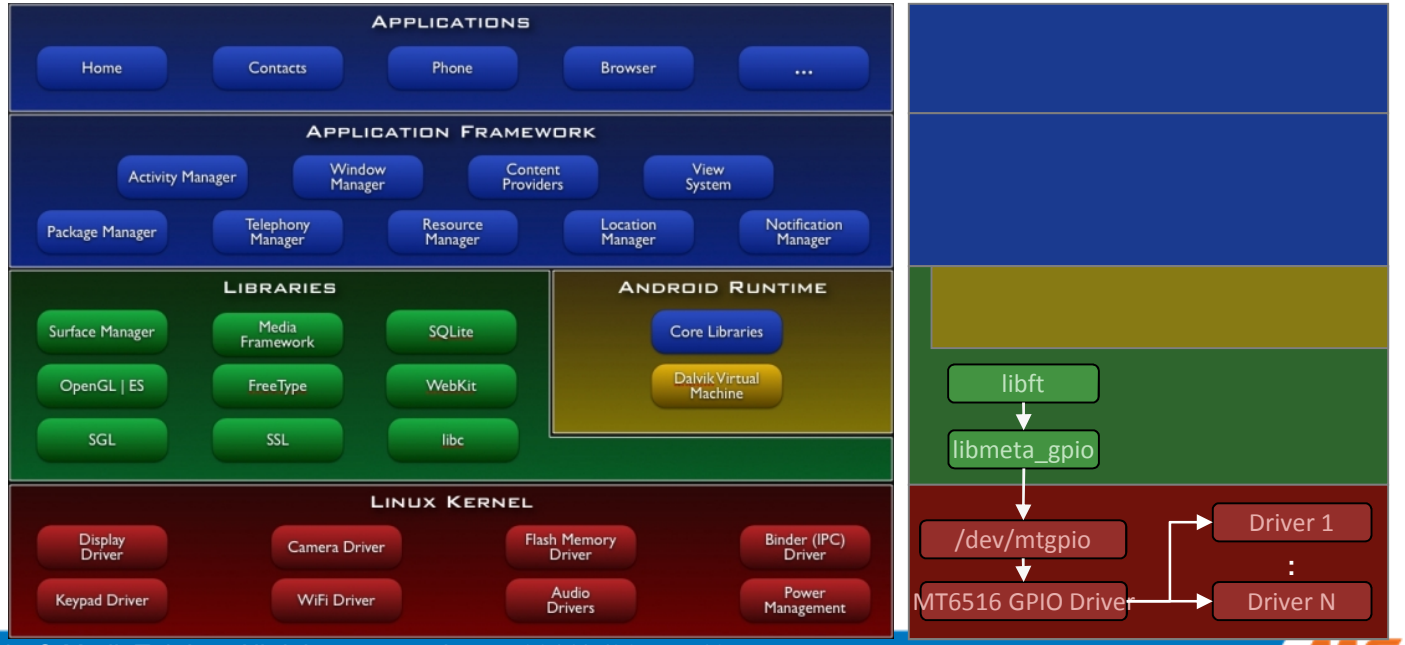
Audio

Misc.
- ***GPIO***
- I2C
- Headset
- PMIC

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

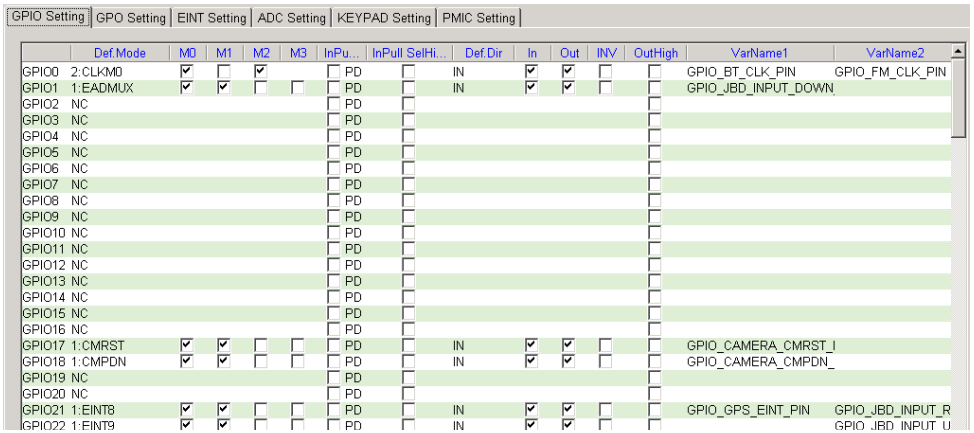
GPIO Introduction

- MT6573 has 147 GPIO pins, each pin could be configured separately
- The GPIO pin settings are separated into two phases
  - Bootloader phase configures necessary GPIO pins
  - Kernel phase configures all GPIO pins to default value set by customization tool



GPIO Customization (1/2)

- DCT tool



- cust_gpio_boot.h defined the default value of all GPIO pins

```
//Configureation for Pin 4
#define GPIO4_MODE      GPIO_MODE_DEFAULT
#define GPIO4_DIR       GPIO_DIR_IN
#define GPIO4_PULLEN    GPIO_PULL_ENABLE
#define GPIO4_PULL      GPIO_PULL_DOWN
#define GPIO4_DATAOUT   GPIO_OUT_ZERO
#define GPIO4_DATAINV   GPIO_DATA_UNINV
```

- cust_gpio_usage.h is used to customize the gpio pins in drivers

```
#define GPIO_CAMERA_CMRST_PIN         GPIO17
#define GPIO_CAMERA_CMRST_PIN_M_GPIO   GPIO_MODE_00

#define GPIO_CAMERA_CMPDN_PIN         GPIO18
#define GPIO_CAMERA_CMPDN_PIN_M_GPIO   GPIO_MODE_00
```

GPIO Customization (2/2)

- Change file list

File Name	Location
cust_gpio_boot.h	alps\mediatek\source\custom\${BOARD}\kernel\dct\dct\
cust_gpio_usage.h	alps\mediatek\source\custom\${BOARD}\kernel\dct\dct\
board.c	alps\mediatek\source\custom\${BOARD}\kernel\core\src\
gpio_init.c	alps\kernel\arch\arm\mach-mt6573\

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- JNI
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

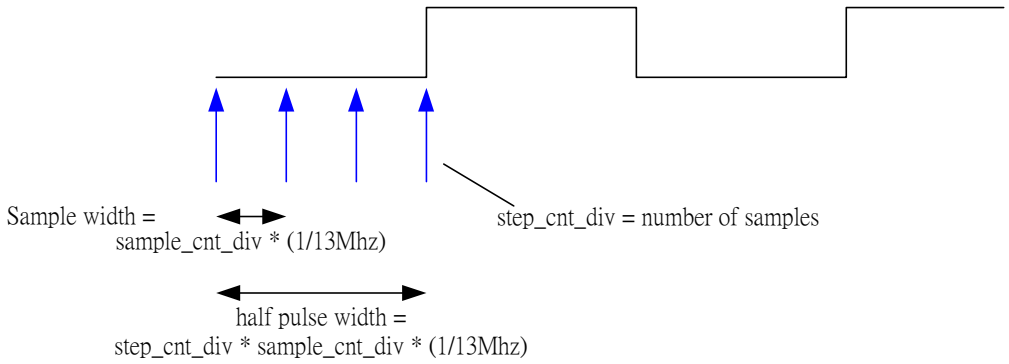
Audio

Misc.
- GPIO
- ***I2C***
- Headset
- PMIC

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

Programmable Timing Parameters of I2C

- The system sets default parameters for I2C.
- Some I2C slave devices need to meet their timing requirements.



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name						**SAMPLE_CNT_DIV**					**STEP_CNT_DIV**					
Type						R/W					R/W					
Reset						‘h3					‘h3					

Customize APIs

- Two series APIs in the I2C driver perform master send/receive transactions.
  - i2c_master_send(), i2c_master_recv()
  - i2c_transfer()
- Customization of I2C APIs is **ONE-SHOT only**.
  - I2C_A_FILTER_MSG : filter out error message
  - I2C_DMA_FLAG : enable DMA transaction
  - I2C_WR_FLAG: enable write and read transaction
  - I2C_RS_FLAG: enable repeat start transaction

Notification

- MT6516 I2C feature
  - i2c0 /i2c1 is dedicate to camera /pmic independent;
  - Only i2c2 can used to porting peripheral device;
  - i2c2 don’t support DMA, so one-time transfer can’t exceed 8 bytes

- MT6573 I2C feature
  - Both i2c0 and i2c1 support DMA

Transfer examples

- Filter out error message
  - client->addr = SLAVE_ADDR;
  - client->addr|= *I2C_A_FILTER_MSG*;
  - i2c_master_recv(client, buf, count);
- Adjust timing
  - client->addr = SLAVE_ADDR;
  - client->timing = 400; // means 400 kps
  - i2c_master_recv(client, buf, count);
- Write-Read and Repeat start
  - client->addr= SLAVE_ADDR & I2C_MASK_FLAG;
  - client->addr|= I2C_WR_FLAG | I2C_RS_FLAG;
  - i2c_master_send(client, buf, read_count<<8| write_count);

DMA example

- static u8 *gpDMABuf_va = NULL;
- static u32 gpDMABuf_pa = NULL;
- gpDMABuf_va = (u8 *)dma_alloc_coherent(NULL, 4096, &gpDMABuf_pa, GFP_KERNEL);
- new_client->addr = new_client->addr & I2C_MASK_FLAG | I2C_DMA_FLAG;
- gpDMABuf_va[0] = cmd;
- i2c_master_send(new_client, gpDMABuf_pa , write_length);

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

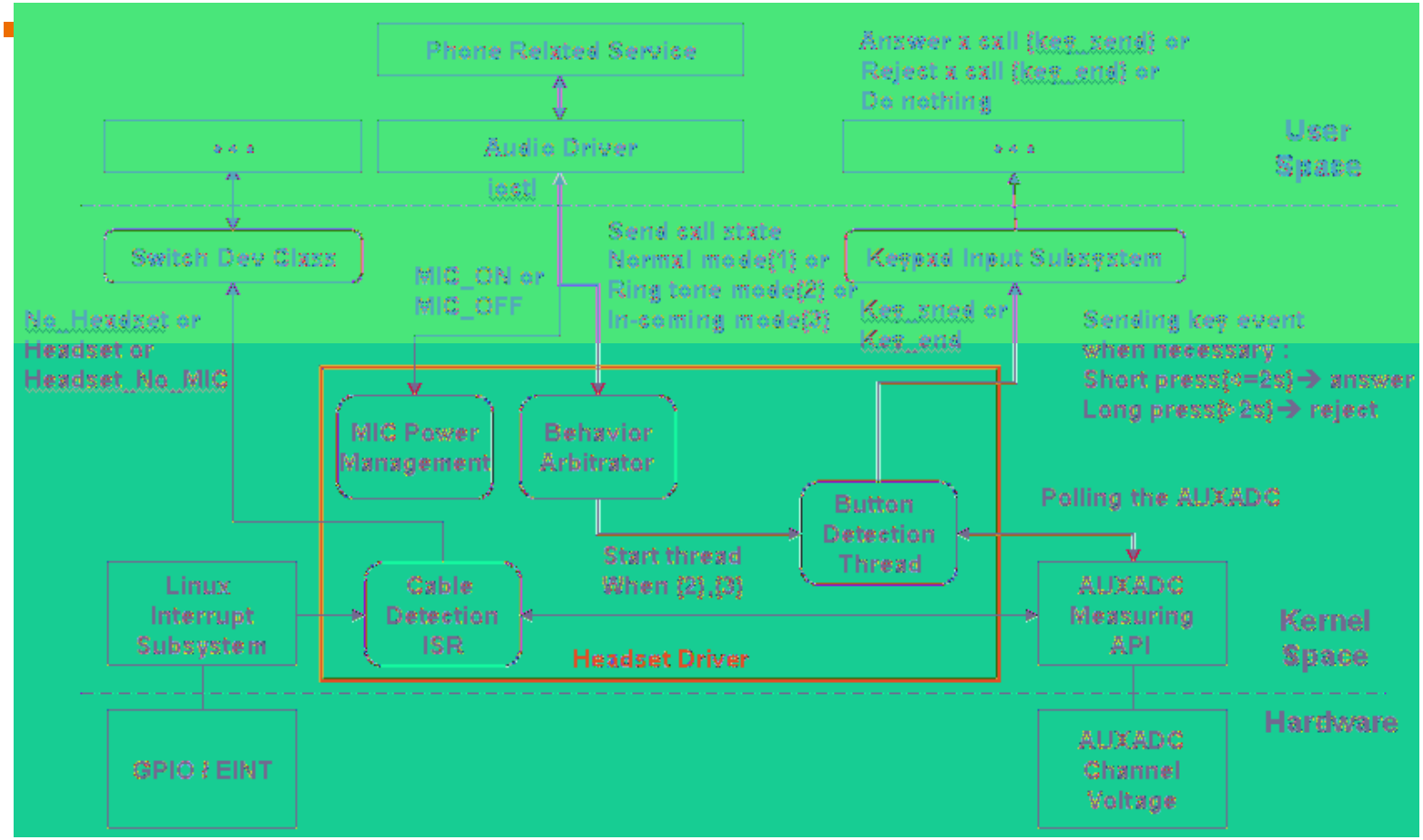
Sensor System
- Sensor Hal
- Sensor Driver Customization

Audio

Misc.
- GPIO
- I2C
- ***Headset***
- PMIC

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

Headset Introduction --MT6516



Headset Customization

- Change file list

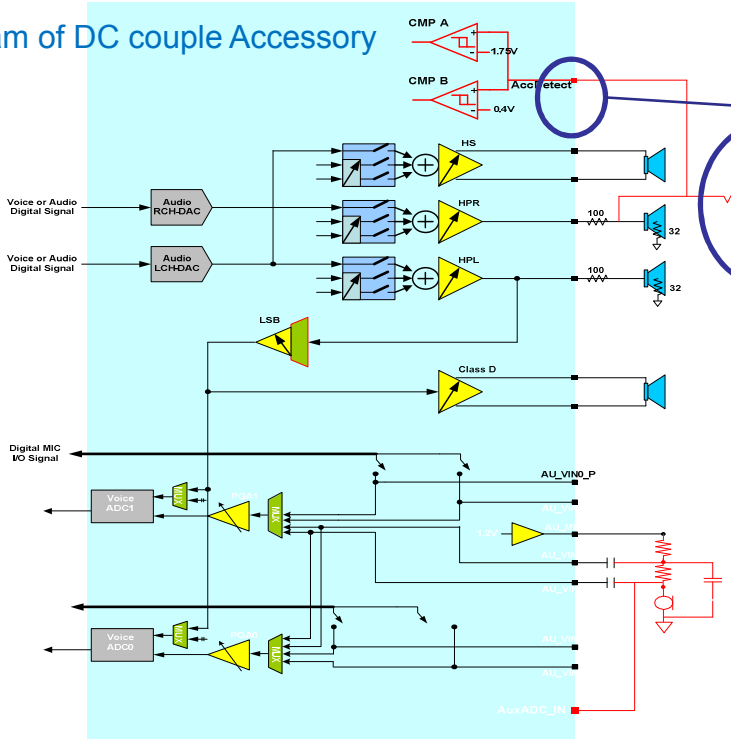
File	Description
\alps\kernel\sound\mtksound\mt6516\headset	
yusu_android_headset.c	The implementation of headset related APIs.
\alps\kernel\sound\yususound	
Headset_custom.h	The headset related settings.

- Customization item
  - CUST_EINT_HEADSET_NUM
    - Please use DCT – EINT for configuring the external interrupt for the headset.
  - GPIO_HEADSET_INSERT_PIN
    - Please use DCT – GPIO for configuring the GPIO line for the headset.
  - Cable_insert_level
    - Headset_custom.h
  - Adc_channel
    - Headset_custom.h

```
bool cable_insert_level = 0; //cable insert level.

//remote button customization
#ifdef HEADSET_RM_ENABLE
int long_press_time = 2000;//the time that press remote
#endif

#ifdef USE_ADC
//ADC customization
char adc_channel = 5;
```

Headset Introduction --MT6573

Block Diagram of DC couple Accessory detect



Connect ACCDET to Audio Buffer output

External circuit to generate DC bias at AVDD/2

Interrupt

ACCDET'S Register

Headset/
TV OUT/
Button Press
……

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

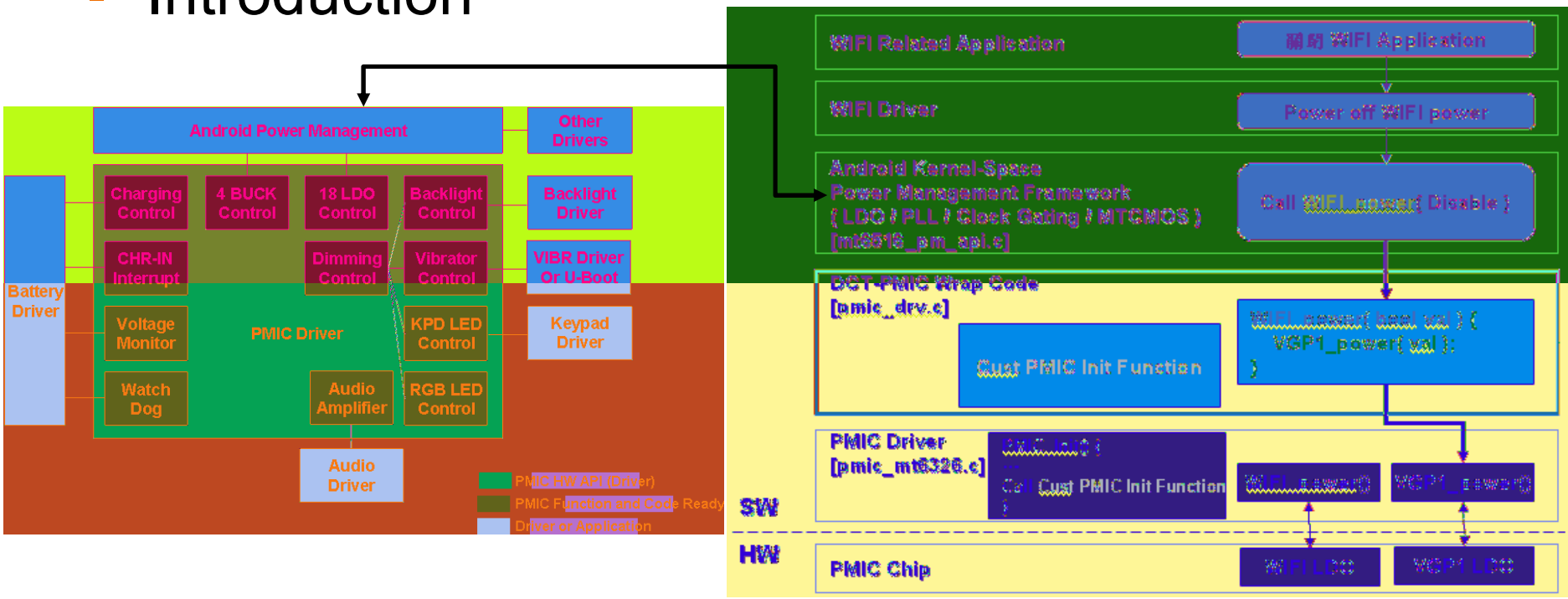
Audio

Misc.
- GPIO
- I2C
- Headset
- ***PMIC***

Appendix
- OFN
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

PMIC Introduction

- Introduction



Remind: MT6516 not support PMIC DCT

PMIC Customization

- Change file list

File	Description
alps\mediatek\platform\mt6516\kernel\drivers\power	
pmic_mt6326.c	The implementation of the PMIC related APIs.
alps\mediatek\platform\mt6573\kernel\drivers\power	
pmu6573.c	The implementation of the PMIC related APIs.
alps\mediatek\custom\${BOARD}\kernel\dct\dct	
pmic_drv.c	The LDO setting APIs
alps\mediatek\custom\${BOARD}\kernel\dct\dct	
pmic_drv.h	The header file for the LDO setting APIs

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

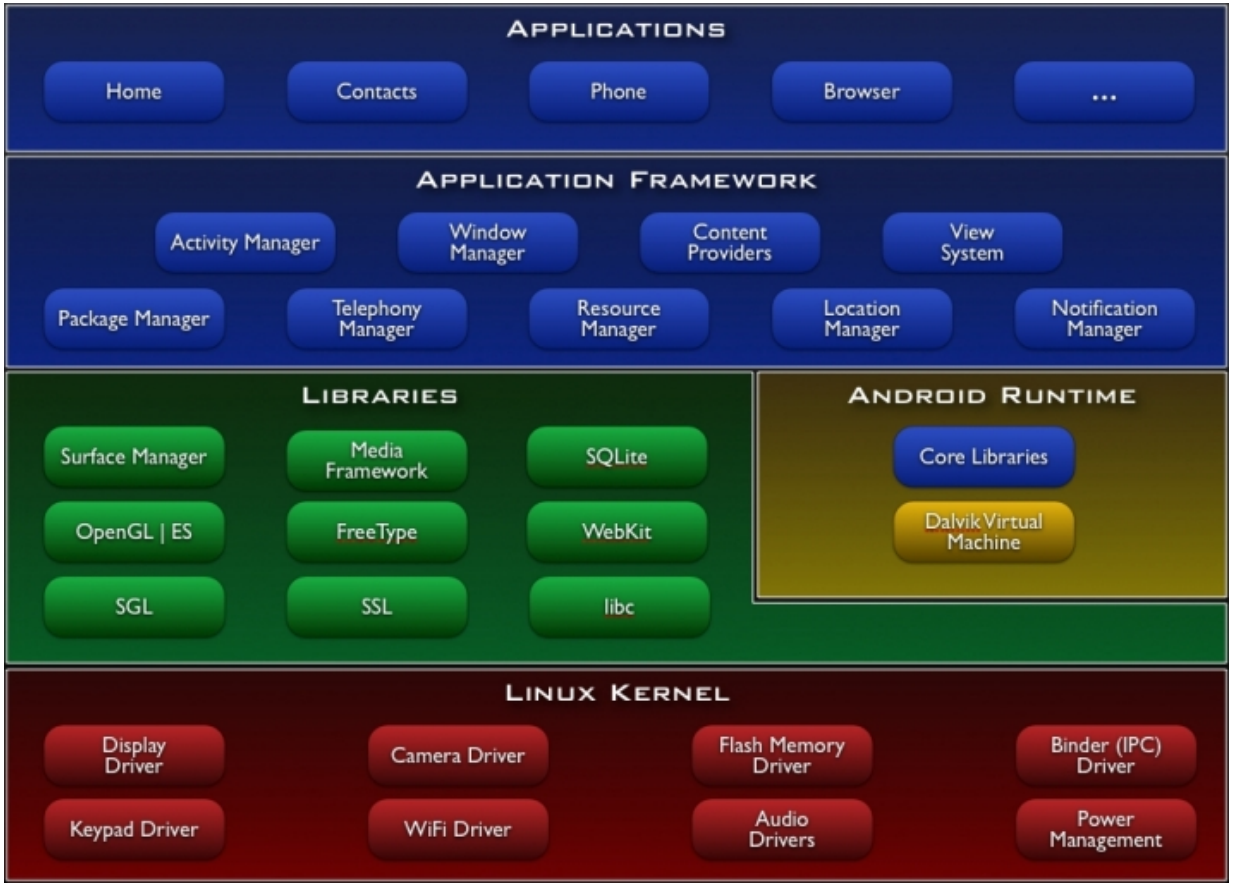
Sensor System
- Sensor Hal
- Sensor Driver Customization

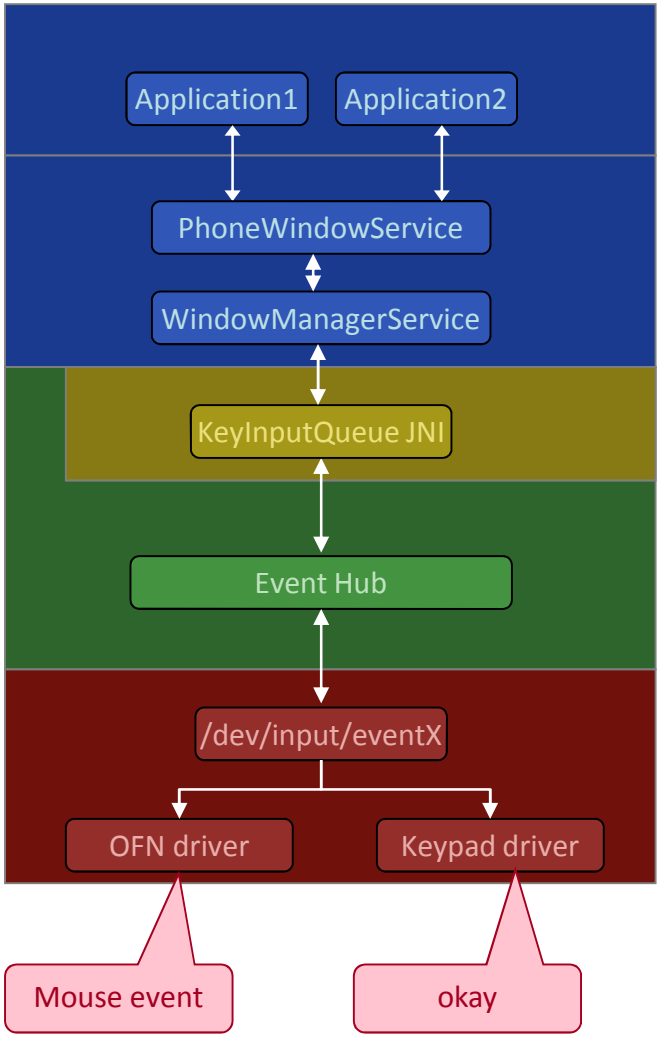
Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix
- ***OFN***
- FM
- BT
- RTC
- Modem
- Jogball
- SDHC
- Camera
- Factory
- USB
- GPS
- LCM Porting
- Recovery

OFN Introduction





OFN Customization (1/4)

- Change file list

File Name	Location
cust_ofn.h	Android 2.2 alps\mtk\src\custom\common\kernel\ofn\inc Android 2.3 alps\mediatek\custom\common\kernel\ofn\inc
cust_ofn.c	Android 2.2 alps\mtk\src\custom\${BOARD}\kernel\ofn\${MODULE} Android 2.3 alps\mediatek\custom\${BOARD}\kernel\ofn\${MODULE}

- Customization item
  - Overview

```
struct ofn_hw {
    int             power_id;
    int             power_vol;

    int             report_cls;
    OFN_ID          chip_id;
    int             slave_addr;
    int             i2c_num;
    unsigned int    layout;

    /*trackball class*/
    int             quan_x;
    int             quan_y;
    int             accu_max;

    /*keyboard class*/
            :
};
```

OFN Customization (2/4)

- power_id / power_vol
  - Customer could define power source of device according to layout
  - Please refer to the following file for power id and voltage
    - arch\arm\mach\include\mach-mt6516\include\mach\mt6516_pll.h
  - If the power source can't be shutdown, please set the power_id as MT6516_POWER_NONE
- report_cls
  - Since Android expects OFN acts as a mouse, please set it as OFN_CLASS_TRACKBALL
- chip_id
  - For different model, the initialization sequence differs. Please choose the correct chip id, or the OFN could not be correctly enabled
- slave_addr
  - The i2c slave address depends on layout. Please fill the correct i2c slave address in different platform

OFN Customization (3/4)

- i2c_num
  - Customer can define the I2C number used by OFN
  - The value could be defined as 0 ~ 2
- layout
  - The data reported from register will vary depends on device layout. The field is a three bit binary. Please see the following definition:

BIT	Field	Description
BIT 2	XY_SWAP	0 = Normal sensor reporting of DX, DY (**default**) 1 = Swap data of DX to DY and DY to DX
BIT 1	Y_INV	0 = Normal sensor reporting of DY. (**default**) 1 = Invert data of DY only
BIT 0	X_INV	0 = Normal sensor reporting of DX. (**default**) 1 = Invert data of DX only

- quan_x / quan_y
  - The quatized step for x/y axis movement
  - To adjust the sensitivity
- accu_max
  - The maximum accumulated count in each motion interrupt
  - To suppress large motion

OFN Customization (4/4)

- DCT Customization

DCT definition	Description
GPIO_OFN_DWN_PIN	Shutdown pin. It's important in power on / shutdown sequence
GPIO_OFN_RST_PIN	Reset pin. It's important for power on sequence
GPIO_OFN_EINT_PIN	The external interrupt pin for detecting motion
CUST_EINT_OFN_NUM	The ID of external interrupt used for OFN
CUST_EINT_OFN_DEBOUNCE_CN	The debounce count of EINT pin. It's set as zero by default.
CUST_EINT_OFN_POLARITY	The polarity of EINT pin. It's set as low by default.
CUST_EINT_OFN_SENSITIVE	The sensitivity of EINT pin. It's set as level sensitive by default
CUST_EINT_OFN_DEBOUNCE_EN	The debounce enable of EINT pin. It's set as disable by default

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

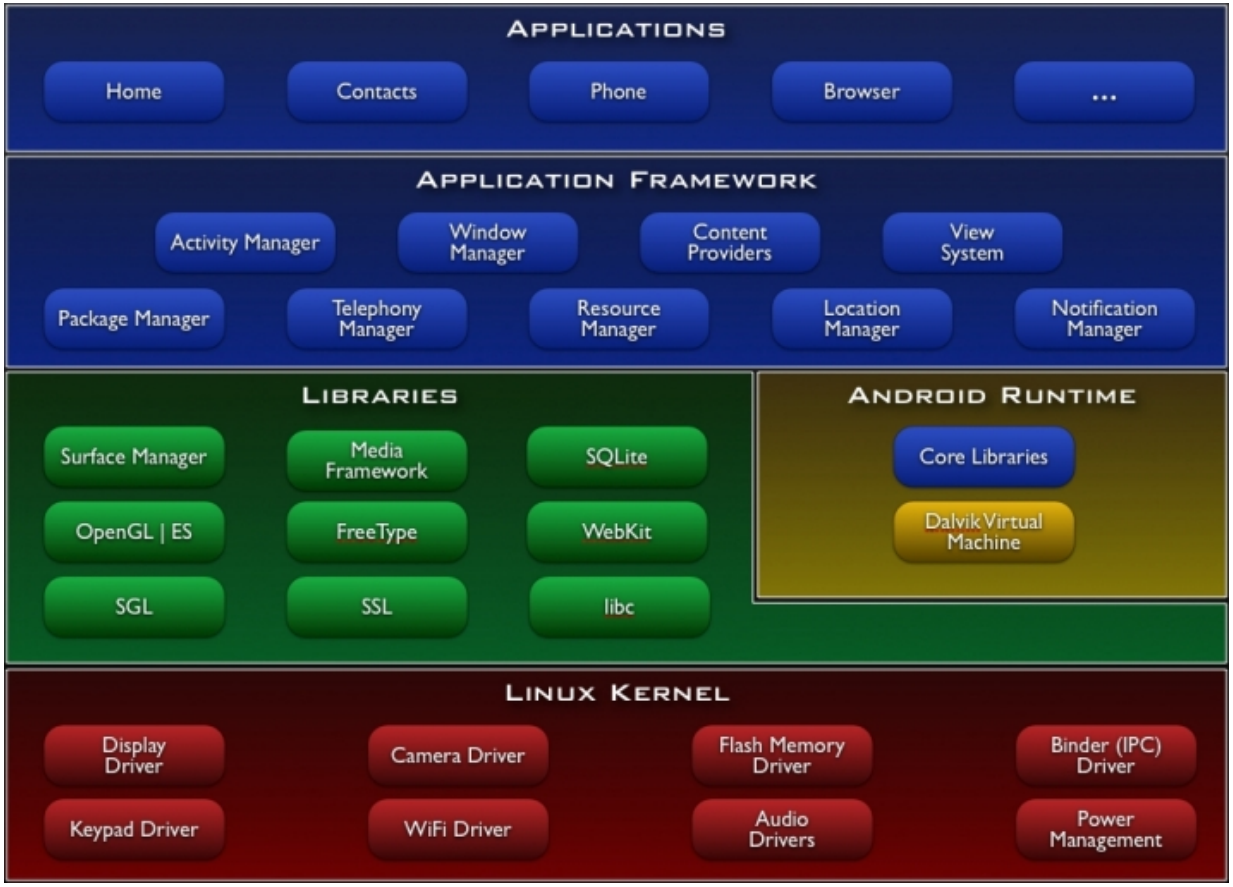
Audio

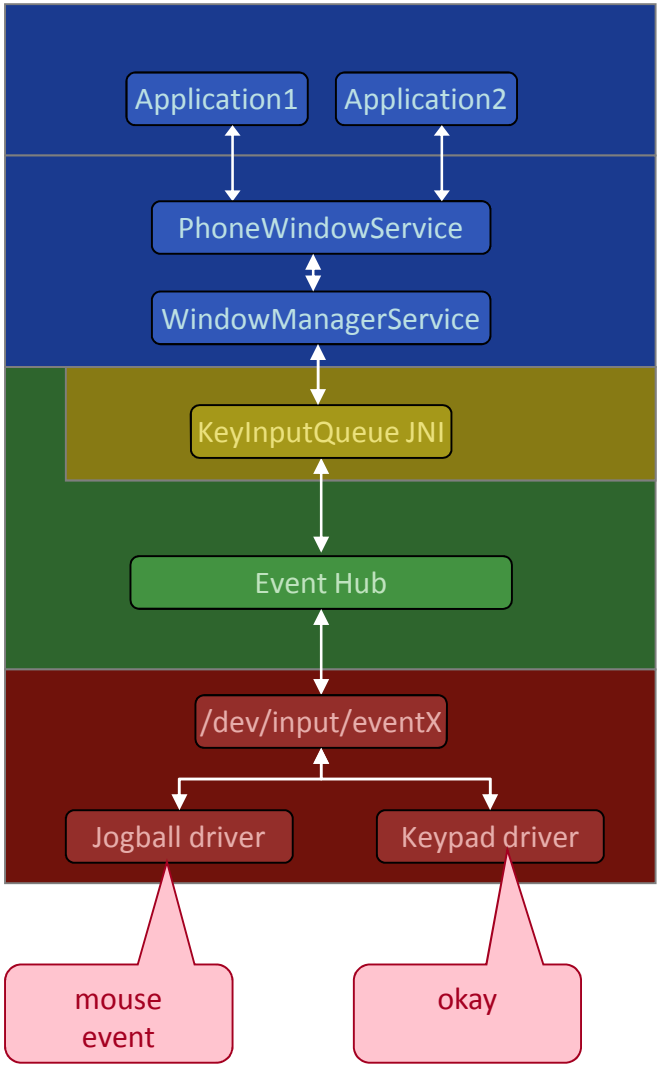
Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix

• OFN	***Jogball***	USB
• FM	SDHC	GPS
• BT	Camera	LCM Porting
• RTC	Factory	Recovery
• Modem		

Jogball Introduction





Jogball Customization (1/3)

- Change file list

File Name	Location
cust_jogball.h	Android 2.2 alps\mtk\src\custom\common\kernel\jogball\inc Android 2.3 alps\mediatek\custom\common\kernel\jogball\inc
cust_jogball.c	Android 2.2 alps\mtk\src\custom\${BOARD}\kernel\jogball\${MODULE} Android 2.3 alps\mediatek\custom\${BOARD}\kernel\jogball\${MODULE}

- Customization item
  - Overview

```
struct jogball_hw {
    int report_cls; /*refer to JBD_CLASS*/

    /*trackball class*/
    int gain_x;
    int gain_y;

    /*keyboard class*/
            :
};
```

  - report_cls
    - Since Android expects jogball acts as a mouse, please set it as JBD_CLASS_TRACKBALL.

Jogball Customization (2/3)

- gain_x / gain_y
  - They are the gain of x-axis and y-axis when detecting one movement
  - The value is currently set as 1 to keep the highest sensitivity

- DCT Customization

DCT definition	Description
GPIO_JBD_INPUT_UP_PIN	The GPIO pin corresponding to up EINT.
GPIO_JBD_INPUT_LEFT_PIN	The GPIO pin corresponding to left EINT.
GPIO_JBD_INPUT_RIGHT_PIN	The GPIO pin corresponding to right EINT
GPIO_JBD_INPUT_DOWN_PIN	The GPIO pin corresponding to down EINT
CUST_EINT_HALL_1_NUM	The ID of up EINT
CUST_EINT_HALL_1_DEBOUNCE_CN	The debounce count, it will generally set as **0x01**
CUST_EINT_HALL_1_POLARITY	The polarity of external pin. It will be set as low level initially. Since any change between 0 and 1 means movement in jogball device, the polarity will be changed during runtime.
CUST_EINT_HALL_1_SENSITIVE	The sensitivity external pin. It will generally set as **edge sensitive** to detect the change between 0 and 1
CUST_EINT_HALL_1_DEBOUNCE_EN	Enable or Disable debounce. It will generally set as **enable** (1)

Jogball Customization (3/3)

- DCT Customization (cont.)

DCT definition	Description
CUST_EINT_HALL_2_NUM	The ID of left EINT
CUST_EINT_HALL_2_DEBOUNCE_CN	The debounce count, it will generally set as **0x01**
CUST_EINT_HALL_2_POLARITY	The polarity of external pin. It will be set as low level initially. Since any change between 0 and 1 means movement in jogball device, the polarity will be changed during runtime.
CUST_EINT_HALL_2_SENSITIVE	The sensitivity external pin. It will generally set as **edge sensitive** to detect the change between 0 and 1
CUST_EINT_HALL_2_DEBOUNCE_EN	Enable or Disable debounce. It will generally set as **enable** (1)
CUST_EINT_HALL_3_NUM	The ID of right EINT
CUST_EINT_HALL_3_DEBOUNCE_CN	The debounce count, it will generally set as **0x01**
CUST_EINT_HALL_3_POLARITY	The polarity of external pin. It will be set as low level initially. Since any change between 0 and 1 means movement in jogball device, the polarity will be changed during runtime.
CUST_EINT_HALL_3_SENSITIVE	The sensitivity external pin. It will generally set as **edge sensitive** to detect the change between 0 and 1
CUST_EINT_HALL_3_DEBOUNCE_EN	Enable or Disable debounce. It will generally set as **enable** (1)
CUST_EINT_HALL_4_NUM	The ID of down EINT
CUST_EINT_HALL_4_DEBOUNCE_CN	The debounce count, it will generally set as **0x01**
CUST_EINT_HALL_4_POLARITY	The polarity of external pin. It will be set as low level initially. Since any change between 0 and 1 means movement in jogball device, the polarity will be changed during runtime.
CUST_EINT_HALL_4_SENSITIVE	The sensitivity external pin. It will generally set as **edge sensitive** to detect the change between 0 and 1
CUST_EINT_HALL_4_DEBOUNCE_EN	Enable or Disable debounce. It will generally set as **enable** (1)

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI
- MT6620 Customization

Sensor System
- Sensor Hal
- Sensor Driver Customization

Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

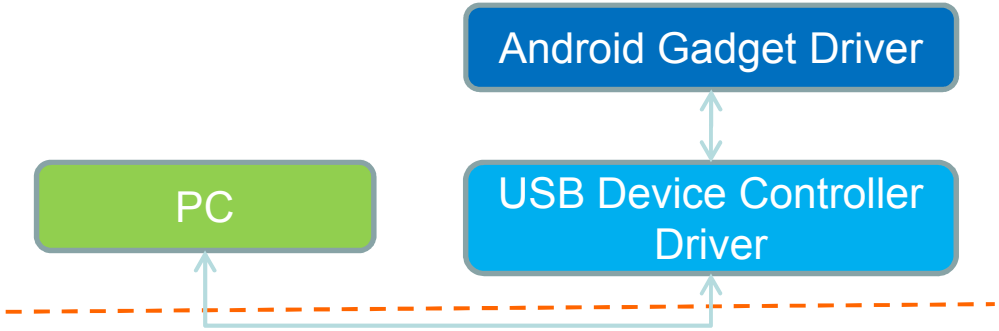
Appendix
- OFN
- FM
- BT
- RTC
- Modem

Jogball
SDHC
Camera
Factory

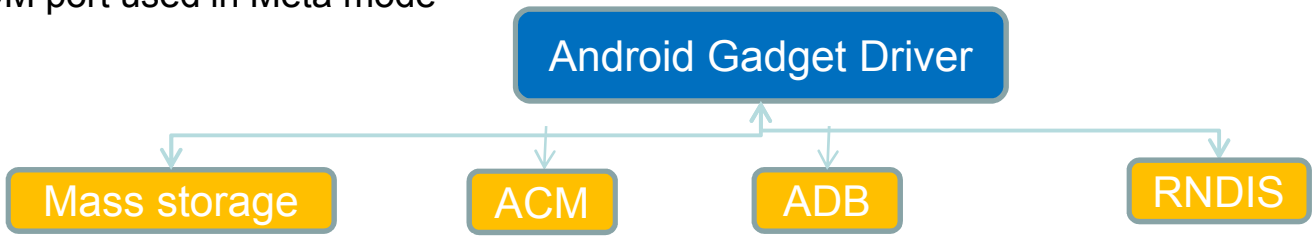
USB
GPS
LCM Porting
Recovery

Overview

- Android USB device mode architecture



- Inside Android gadget driver, there are two interfaces
  - Mass storage interface
    - No driver installation is needed for mass storage function
  - Android Debug Bridge interface
    - Driver installation is needed for this function to work properly
  - RNDIS
  - ACM
    - A virtual COM port used in Meta mode



Customization Items

- CHR_TYPE_DETECT_DEB
  - Debounce time after charger in interrupt is detected before charger type detection in ms

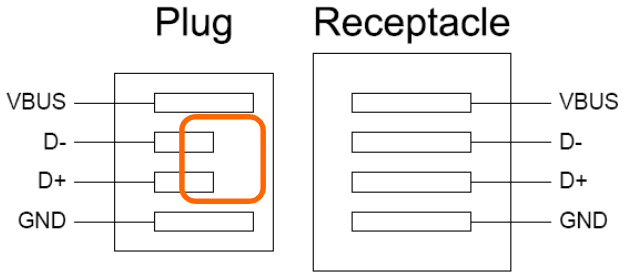
- Change file list

File Name	Location
MTK_usb_custom.h	Android 2.2 alps\mtk\src\custom\common\kernel\usb\usb Android 2.3 Alps\mediatek\custom\mt6573\kernel\usb\src

```
#define USB_MS_PRODUCT_ID         0x0001
#define USB_MS_ADB_PRODUCT_ID     0x0c03
#define USB_RNDIS_PRODUCT_ID      0x0003
#define USB_RNDIS_ADB_PRODUCT_ID  0x0004

#define VENDOR_ID       0x0bb4                  /* USB vendor id  */
#define PRODUCT_ID      USB_MS_PRODUCT_ID       /* USB default product id */

#define MANUFACTURER_STRING "MediaTek"
#define PRODUCT_STRING      "MT65xx Android Phone"

#define USB_ETH_VENDORID      0
#define USB_ETH_VENDORDESCR   "MediaTek"

#define USB_MS_VENDOR         "MediaTek"
#define USB_MS_PRODUCT        "MT65xx MS"
#define USB_MS_RELEASE        0x0100

#define CHR_TYPE_DETECT_DEB   400  /* debounce time for charger type detection, in ms */
```

Customization

- **CHR_TYPE_DETECT_DEB**



- In the process of charger insertion, VBUS/GND is connected with receptacle first, where D+/D- pins are connected later.
- The latency between VBUS/GND connection and D+/D- connection is determined by insertion speed.
- If charger type detection is done before D+/D- pins have been connected, then charger type will be recognized as non-standard charger.
- For this reason, charger type detection debounce is used to mitigate this issue. 400ms is recommended from our experience, but for flexibility, this time can be customized.

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

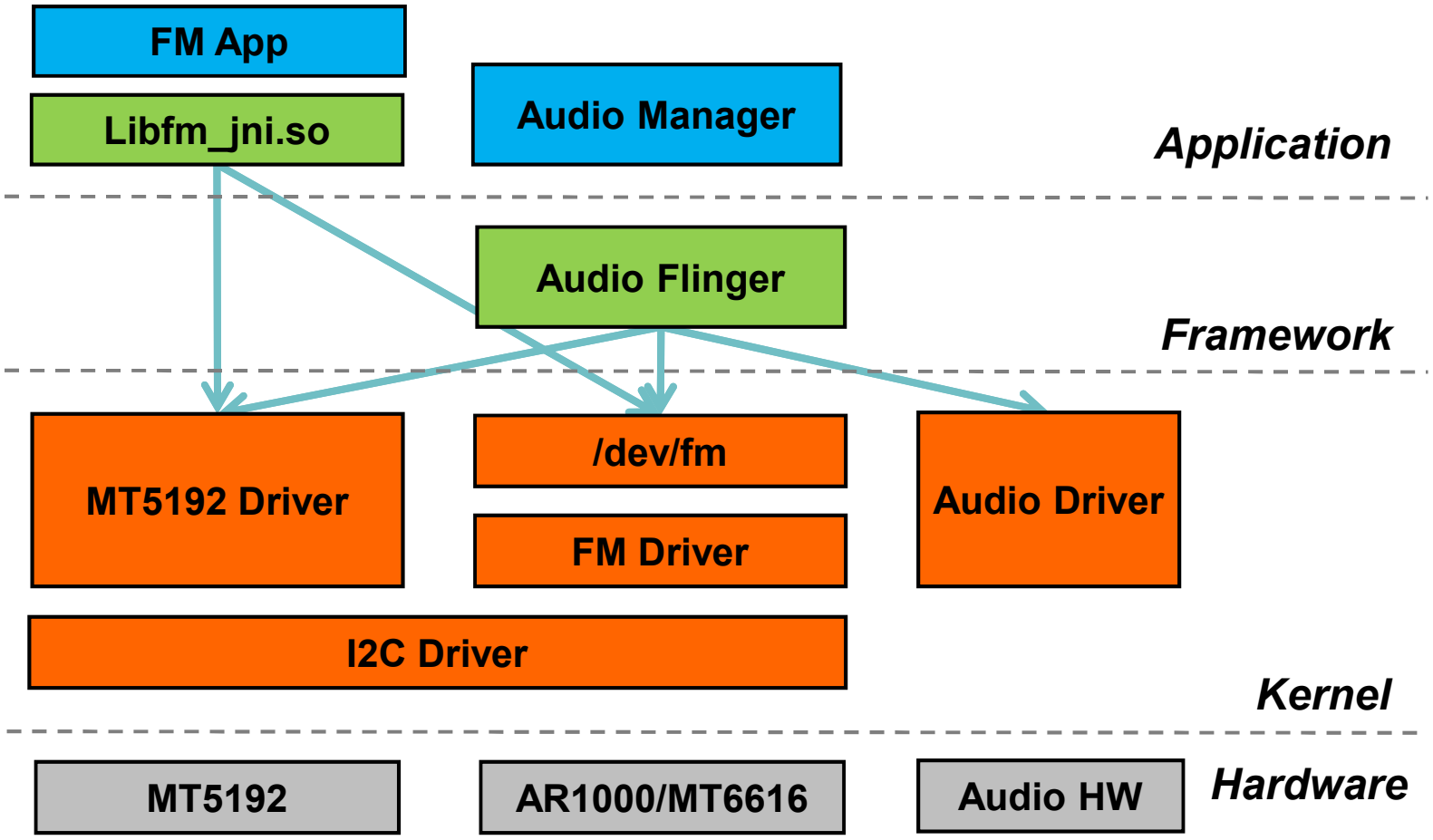
Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix
- OFN
- SDHC
- Camera
- Factory
- Jogball
- GPS
- LCM Porting
- Recovery
- USB
- BT
- RTC
- Modem
- ***FM***

FM Chip Selection

- Now we support following chip driver included in Yusu codebase:
  - AR1000 FM single chip
  - MT6616 FM+BT combo
  - MT5192 FM+ATV combo
  - MT6620 FM(Tx/Rx)+BT+WIFI+GPS combo
- If you need our support to add other chip, pls contact SWPM.
- Makefile configuration

	AR1000	MT6616	MT5192
MTK_FMRADIO_APP	yes	yes	yes
MTK_FM_CHIP	AR1000_FM	MT6616_E3_FM	MT5192_FM
CUSTOM_KERNEL_FM	ar1000	mt6616	
MTK_MT5192_FM_SUPPORT	no	no	yes
Note. For MT5192 FM, pls also check: CUSTOM_KERNEL_MATV = mt5192			


FM App
Libfm_jni.so
Audio Manager
Application
Audio Flinger
Framework
MT5192 Driver
/dev/fm
FM Driver
Audio Driver
I2C Driver
Kernel
MT5192
AR1000/MT6616
Audio HW
Hardware

FM Overview

- YuSu FM driver is a character device driver
  - Provide all functionalities by IOCTL (I/O control)
  - Control the AR1000/MT6616 chip via I2C driver
  - User can control the FM chip via /dev/fm device node (except MT5192)
  - As a tiny interface to provide hardware functions
- FM driver functionalities
  - FM chip initialization
  - Power up/down
  - Tune
  - Seek forward/backward
  - Mute on/off
  - Get/Set volume
  - Get RSSI

GPIO setting

- Power/CLK pin please reference to BT setting
- /alps/mtk/src/custom/$(PROJECT)/kernel/dct/dct/cust_gpio-_usage.h
- /alps/mtk/src/custom/$(PROJECT)/kernel/dct/dct/cust_eint.h
  - (Please use DCT tool to generate this file, not modify it directly)

```
#define GPIO_BT_CLK_PIN          GPIO117
#define GPIO_BT_CLK_PIN_M_GPIO    GPIO_MODE_00
#define GPIO_BT_CLK_PIN_M_CLK    GPIO_MODE_01
#define GPIO_BT_CLK_PIN_CLK      CLK_OUT2
#define GPIO_BT_CLK_PIN_FREQ     CLK_SRC_F32K

#define GPIO_FM_CLK_PIN          GPIO117
#define GPIO_FM_CLK_PIN_M_GPIO    GPIO_MODE_00
#define GPIO_FM_CLK_PIN_M_CLK    GPIO_MODE_01
#define GPIO_FM_CLK_PIN_CLK      CLK_OUT2
#define GPIO_FM_CLK_PIN_FREQ     CLK_SRC_F32K


#define GPIO_FM_RDS_PIN          GPIO22
#define GPIO_FM_RDS_PIN_M_GPIO    GPIO_MODE_00
#define GPIO_FM_RDS_PIN_M_EINT    GPIO_MODE_01


#define CUST_EINT_FM_RDS_NUM              9
#define CUST_EINT_FM_RDS_DEBOUNCE_CN      0
#define CUST_EINT_FM_RDS_POLARITY         CUST_EINT_POLARITY_LOW
#define CUST_EINT_FM_RDS_SENSITIVE        CUST_EINT_LEVEL_SENSITIVE
#define CUST_EINT_FM_RDS_DEBOUNCE_EN      CUST_EINT_DEBOUNCE_DISABLE
```

← MT6616 don't need it

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

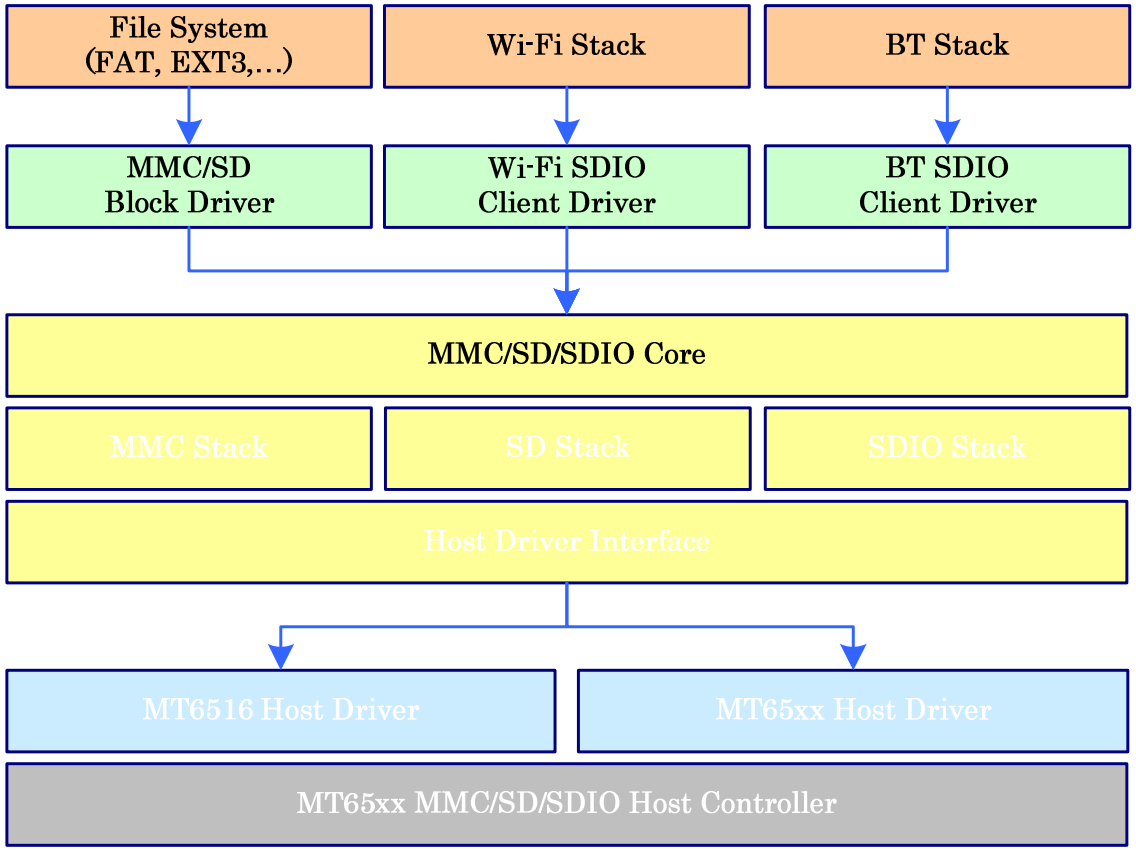
Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix

• OFN	Jogball	USB	FM
• ***SDHC***	GPS	BT	
• Camera	LCM Porting	RTC	
• Factory	Recovery	Modem	

MMC/SD/SDIO Software Architecture



- **MMC/SD Block Driver**
  - A generic block driver to glue the file system and low-level disk/storage driver
- **MMC/SD/SDIO Core**
  - MMC/SD/SDIO card detection, initialization flow, clock, bus width control
  - Power management
- **MMC/SD/SDIO Stack**
  - MMC/SD/SDIO protocol layer implementation
  - Card specified information
- **Host Driver**
  - Handling MMC/SD/SDIO command or data request from subsystem
  - Perform low-level operations to host controller

MMC/SD/SDIO Host Controller Limitation

- Maximum 3 slots for SD memory and SDIO card
- Not support SPI mode for SD/MMC memory card
- SDIO bus interrupt can't be as the wake up source so that an specified EINT is recommended to be used as the SDIO interrupt
- Data bus rate up to 26x4Mbps in parallel mode.

MMC/SD/SDIO Host Customization (1/2)

- Customization Parts
  - Number of MSDC slots on a target
  - Clock source, data pins, driving current,
  - Card detection, write protection, external SDIO interrupt capability
  - External SDIO interrupt handler, external power control handler
  - Host power management policy and manner
- Customization File List
  - `alps/mtk/src/custom/${BOARD}/kernel/core/src`

Customization File	Description
board-custom.h	Customize the available MSDC slots on a target
board.c	Customize host data pins, capability, driving current, clock source, external SDIO interrupt, external card power control, host-specified power management function, etc.

MMC/SD/SDIO Host Customization (2/2)

- Host Customization Example

```
#if defined(CFG_DEV_MSDC2)
struct mt6516_sd_host_hw mt6516_sd2_hw = {
    .clk_src            = MSDC_CLKSRC_MCPLL,
    .cmd_edge           = EDGE_RISING,
    .data_edge          = EDGE_FALLING,
    .cmd_odc            = MSDC_ODC_4MA,
    .data_odc           = MSDC_ODC_4MA,
    .cmd_slew_rate      = MSDC_ODC_SLEW_FAST,
    .data_slew_rate     = MSDC_ODC_SLEW_FAST,
    .data_pins          = 4,
    .data_offset        = 0,
    .flags              = MSDC_EXT_SDIO_IRQ,
    .request_sdio_eirq  = mt5921_wifi_request_irq,
    .enable_sdio_eirq   = mt5921_wifi_enable_irq,
    .disable_sdio_eirq  = mt5921_wifi_disable_irq,
    .register_pm        = mt5921_wifi_register_pm,
};
#endif
```

- clock source
- driving current
- latch timing
- data pins
- user data offset
- capability flags

- external interrupt registration
- external interrupt enable/disable
- power mgmt. callback registration

Capability Flag	Description
MSDC_CD_PIN_EN	Enable MSDC card detection pin
MSDC_WP_PIN_EN	Enable MSDC write protect pin
MSDC_SDIO_IRQ	Enable MSDC SDIO IRQ
MSDC_SDIO_EXT_IRQ	Use external interrupt as SDIO interrupt source
MSDC_SYS_SYSPEND	MSDC is suspended automatically by system

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

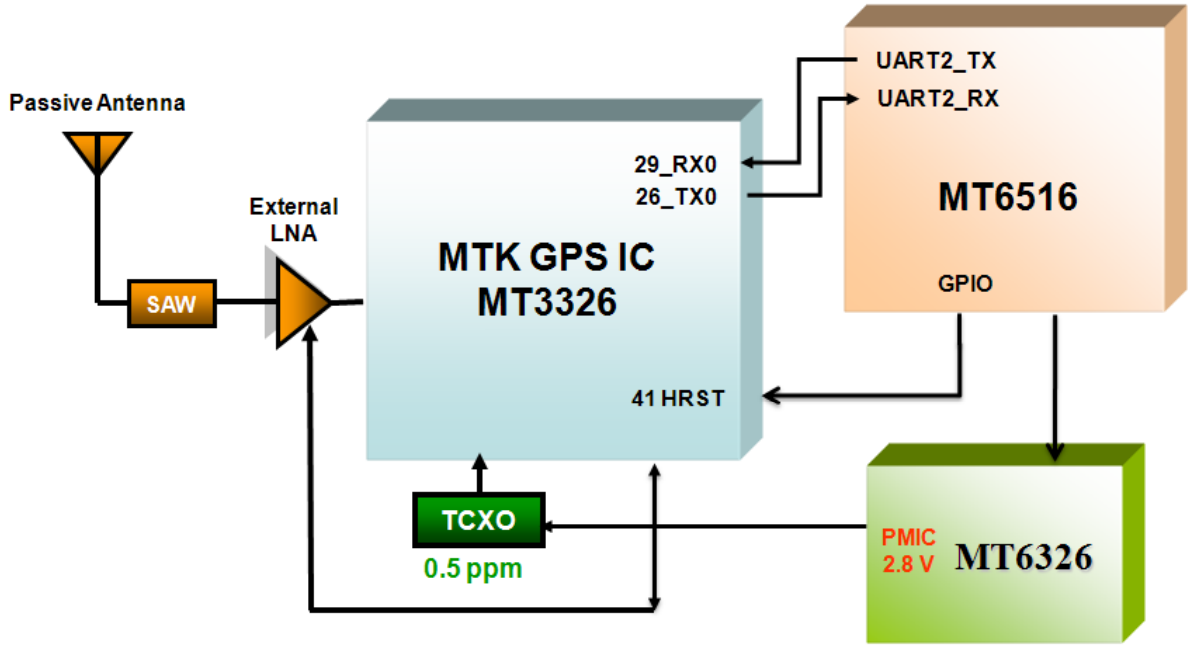
Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix
- OFN
- SDHC
- Camera
- Factory
- Jogball
- ***GPS***
- LCM Porting
- Recovery
- USB
- BT
- RTC
- Modem
- FM

GPS Architecture



GPS customization(1/7)

- **HW Reset Pin**
  - GPS GPIO config in cust_gpio_usage.h
  - H/W Reset pin:
    - **GPIO_GPS_RST_PIN**
  - The PIN settings should be defined with DCT tool.

GPS customization(2/7)

- **Power Control**

Power Control	Customization File	API
MT6326 LDO	kernel\drivers\gps\gps.c	mt3326_gps_power()
External LDO	mtk\src\custom\${BOARD}\kernel\core\src\board.c :	mt3326_gps_ext_power_on() mt3326_gps_ext_power_off()

```
static inline void mt3326_gps_power(struct mt3326_gps_hardware *hw,
                                    unsigned int on, unsigned int force)
{

    if (!hwPowerOn(MT6516_POWER_V3GTX, VOL_2800,"MT3326")) {
        GPS_ERR("power on fails!!\n");
        return;
    }

    if (!hwPowerDown(MT6516_POWER_V3GTX,"MT3326" ))
        GPS_ERR("power off fail\n");
```

MT6326 LDO

GPS customization(3/7)

```
/*GPS driver*/
struct mt3326_gps_hardware mt3326_gps_hw = {
    .ext_power_on =  mt3326_gps_ext_power_on,
    .ext_power_off = mt3326_gps_ext_power_off,
};
```

External LDO

```
int mt3326_gps_ext_power_on(int state)
{
    s32 err = 0;

    if(state == 1)
    {
        /*configure as gpio function*/
        err = mt_set_gpio_mode(GPIO_GPS_PWREN_PIN, GPIO_GPS_PWREN_PIN_M_GPIO);
        if (RSUCCESS != err)
            DEV_ERR("set power fails: %d\n", err);
        /*configure as output*/
        err = mt_set_gpio_dir(GPIO_GPS_PWREN_PIN, GPIO_DIR_OUT);
        if (RSUCCESS != err)
            DEV_ERR("set power fails: %d\n", err);
        /*configure as 1, enable power*/
        err = mt_set_gpio_out(GPIO_GPS_PWREN_PIN, GPIO_OUT_ONE);
        if (RSUCCESS != err)
            DEV_ERR("set power fails: %d\n", err);
```

```
int mt3326_gps_ext_power_off(void)
{
    s32 err;
    err = mt_set_gpio_out(GPIO_GPS_PWREN_PIN, GPIO_OUT_ZERO);
    if (RSUCCESS != err)
        DEV_ERR("set power fails: %d\n", err);
    return err;
}
```

GPS customization(4/7)

- GPS HW related setting modify default value:
  - File:mtk\src\custom\${BOARD}\cgen\cfgdefault\CFG_GPS_Default.h

```
ap_nvram_gps_config_struct stGPSConfigDefault =
{
    /* "/dev/ttyMT1" */
    {'/','d','e','v','/','t','t','y','M','T','1',0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x
    /* 0:s/w, 1:none, 2:h/w */
    1,

    /* 16.368MHz */
    16368000,
    /* 500ppb */
    500,
    /* 0:16.368MHz TCXO */
    0,

    /* 0:mixer-in, 1:internal-LNA */
    0,

    /* 0:none */
    0
};
```

GPS customization(5/7)

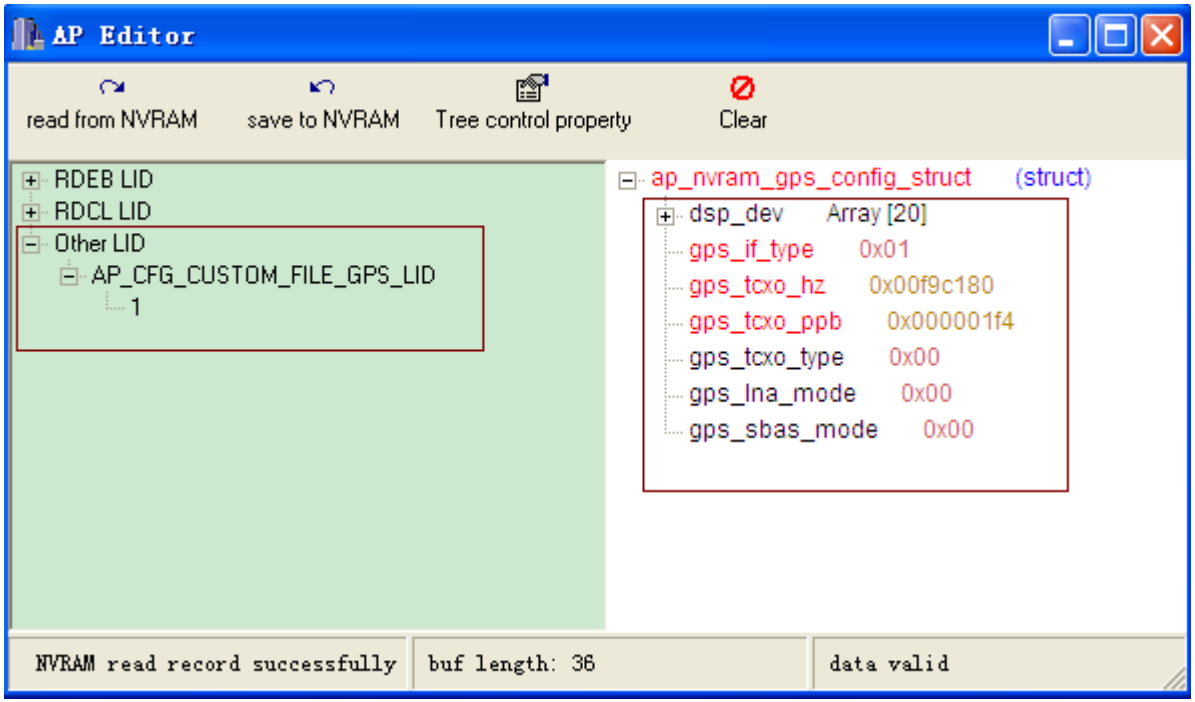
- GPS HW related settings get through NRAM
  - UART settings
    - dsp_dev[20] : defines the TTY path and name to connect to MT3326.
      - Default : {'/','d','e','v','/','t','t','y','M','T','1',0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0} // /dev/ttyMT1
    - gps_if_type : defines whether to use UART flow control or not
      - 0 → S/W flow control
      - 1 → none
      - 2 → use UART H/W flow control, UART RTS / CTS must be connected
      - Default :
        - **1** // no flow control
  - TCXO settings
    - gps_tcxo_type : clock type
      - 0 :TCXO type
      - ff: not precise clock
    - gps_tcxo_hz : TCXO frequency, in HZ unit
      - Default :
      - 16368000 //16.368MHz
    - gps_tcxo_ppb : TCXO drift, in ppb unit
      - Default :
      - 500 //0.5ppm = 500ppb

GPS customization(6/7)

- GPS HW related setting saved in META and can be modified through META tool(cont)
  - LNA settings
    - gps_lna_mode : GPS LNA type
      - 0 → Mixer-in
      - 1 → Internal LNA
      - Default :
      - 0 //mixer-in

GPS customization(7/7)

- GPS HW related setting through NVRAM editor dynamically



Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

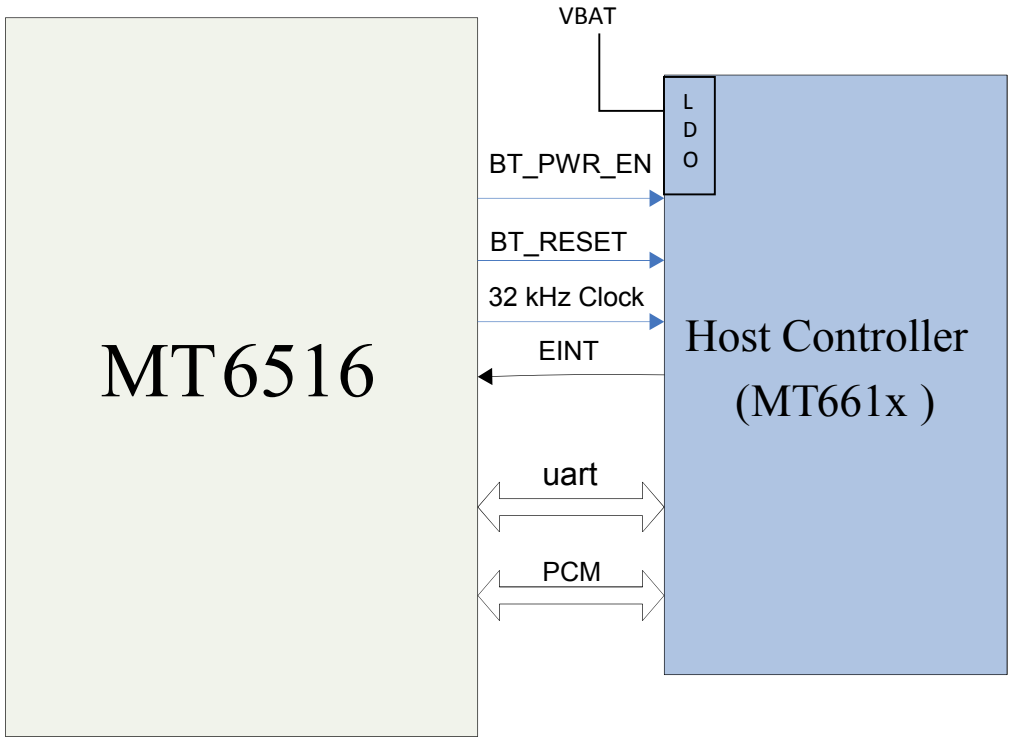
Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix
- OFN
- SDHC
- Camera
- Factory
- Jogball
- GPS
- LCM Porting
- Recovery
- USB
- ***BT***
- RTC
- Modem
- FM

Bluetooth customization(1/5)

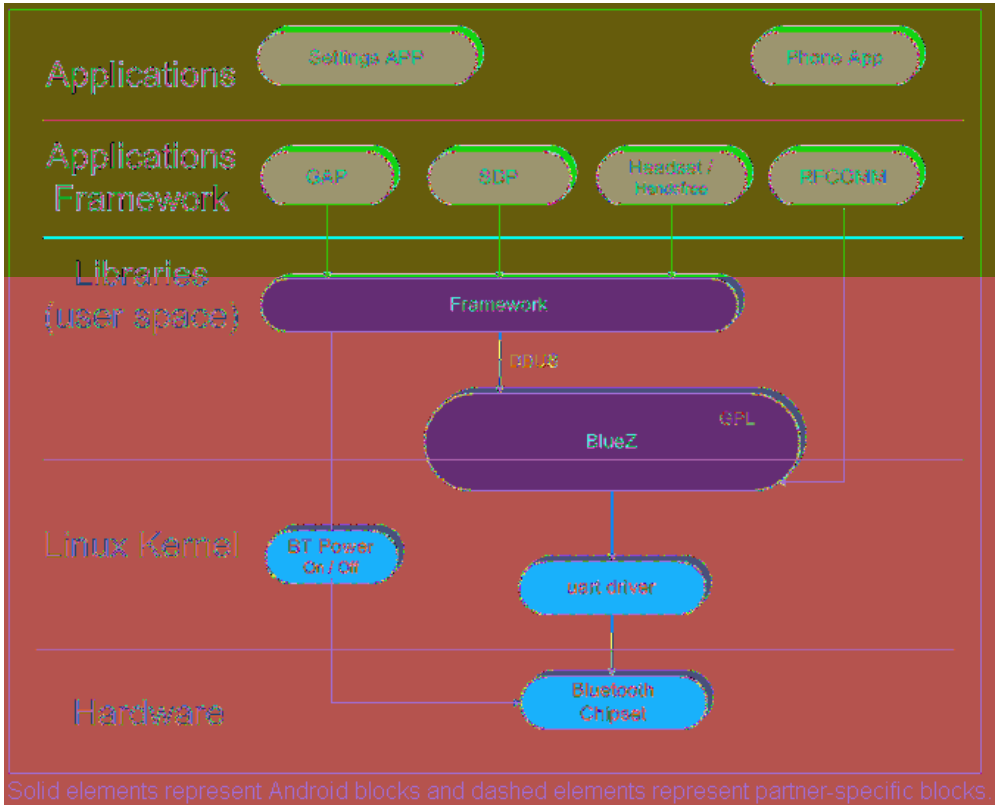
- Bluetooth



Hardware Architecture

Bluetooth customization(2/5)

- BT SW architecture



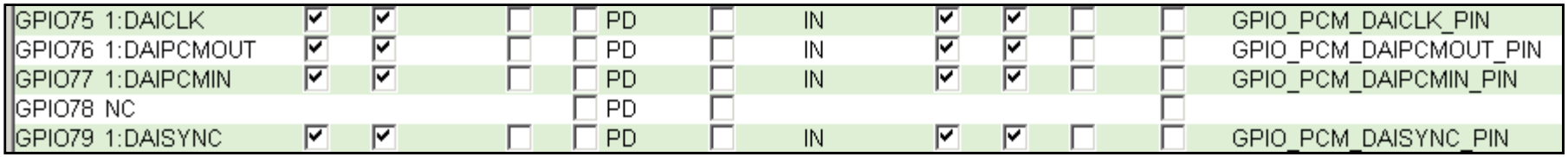
Bluetooth customization(3/5)

- Makefile
  - MTK_BT_CHIP = MTK_MT661x (x=1/2/6)
- BT GPIO config in DCT Tool (cust_gpio_usage.h)
  - PCM:
    - GPIO_PCM_DAICLK_PIN,
    - GPIO_PCM_DAIPCMOUT_PIN,
    - GPIO_PCM_DAIPCMIN_PIN,
    - GPIO_PCM_DAISYNC_PIN
  - External interrupt:
    - GPIO_BT_EINT_PIN
  - 32K Clock source:
    - GPIO_BT_CLK_PIN
  - Power enable and reset pin:
    - GPIO_BT_POWREN_PIN
    - GPIO_BT_RESET_PIN

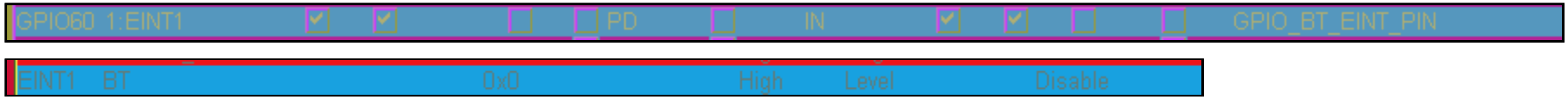
File name	Location
cust_gpio_usage.h	\alps\mtk\src\custom\${BOARD}\kernel\dct\dct
\${BOARD}.mak	\alps\mtk\make

Bluetooth customization(3/5)

– PCM:



– External interrupt:



– 32K Clock source:

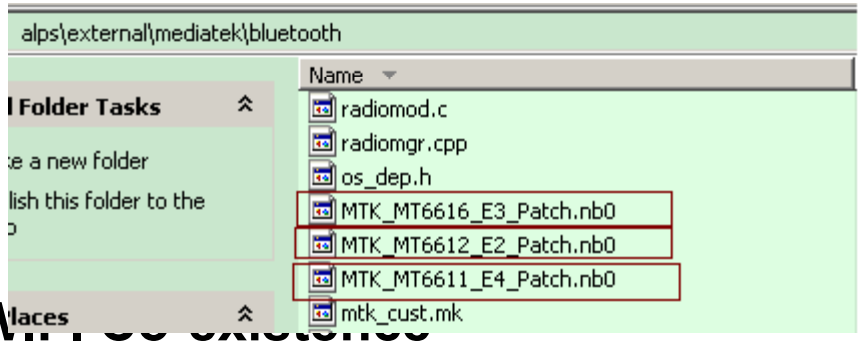
– Power enable and reset pin:

Bluetooth customization(4/5)

- **Bluetooth UART Customization**
  - Announce Bluetooth UART port custom folder
    - File: alps\mtk\make\${BOARD}.mak
    - CUSTOM_HAL_BLUETOOTH = bluetooth
  - UART port and baudrate (not recommended to modify)
    - File:alps\mtk\src\custom\${BOARD}\hal\bluetooth\bluetooth\cust_bt.h
    - #define CUST_BT_SERIAL_PORT "/dev/ttyMT2”
    - #define CUST_BT_BAUD_RATE 3250000
  - At last set the UART port attribute in init.rc (optional)
    - File:alps\build\target\board\${BOARD}\init.rc,
    - # BT
    - chmod 0666 /dev/ttyMT2
    - chown bluetooth bluetooth /dev/ttyMT2

Bluetooth customization(5/5)

- **Bluetooth firmware patch update (implemented by MTK only, do not modify)**
  - Path: alps\external\mediatek\bluetooth
    - MTK_MT66XX_EY_Patch.nb0 (XX means chip, Y means ECO version)



- **BT WIFI co-existence**
  - Pls refer to WIFI part

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix

• OFN	Jogball	USB	FM
• SDHC	GPS	BT	
• ***Camera***	LCM Porting	RTC	
• Factory	Recovery	Modem	

How to change sensor driver for new project

- To do list for change sensor
  - Put sensor driver within custom common part
    - alps\mediatek\custom\common\kernel\imgsensor\${MODULE}
    - alps\mediatek\custom\common\kernel\imgsensor\ov3640_yuv
  - Create project sensor folder and add tuning files
    - alps\mediatek\custom\${BOARD}\hal\imgsensor\${MODULE}
    - alps\mediatek\custom\mt6516_evb\hal\imgsensor\ov3640_yuv
  - Modify project make file
    - File
      - alps\mediatek\config\$project$\ProjectConfig.mk
    - Sample
      - alps\mediatek\config\mt6573_evb\ProjectConfig.mk
      - CUSTOM_HAL_IMGSENSOR = ov3640_yuv
      - CUSTOM_KERNEL_IMGSENSOR = ov3640_yuv
  - Build *kernel* driver and Android system. Then update boot.img and system.img

How to add dual and backup sensor for new project

- To do list for ad dual sensor
  - Modify project make file
    - File
      - alps\mediatek\config\$project$\ProjectConfig.mk
    - Sample
      - alps\mediatek\config\mt6573_evb\ProjectConfig.mk
      - Modify AUTO_ADD_GLOBAL_DEFINE_BY_VALUE
        - Remove CUSTOM_KERNEL_IMGSENSOR
        - add CUSTOM_KERNEL_MAIN_IMGSENSOR
          CUSTOM_KERNEL_MAIN_BACKUP_IMGSENSOR
          CUSTOM_KERNEL_SUB_IMGSENSOR
          CUSTOM_KERNEL_SUB_BACKUP_IMGSENSOR
      - Remove option
        - CUSTOM_HAL_IMGSENSOR = ov3640_yuv
        - CUSTOM_KERNEL_IMGSENSOR = ov3640_yuv
      - Add option
        - CUSTOM_HAL_MAIN_IMGSENSOR = ov3640_yuv
        - CUSTOM_HAL_MAIN_BACKUP_IMGSENSOR =
        - CUSTOM_HAL_SUB_IMGSENSOR = ov7675_yuv
        - CUSTOM_HAL_SUB_BACKUP_IMGSENSOR =
        - CUSTOM_KERNEL_MAIN_IMGSENSOR = ov3640_yuv
        - CUSTOM_KERNEL_MAIN_BACKUP_IMGSENSOR =
        - CUSTOM_KERNEL_SUB_IMGSENSOR = ov7675_yuv
        - CUSTOM_KERNEL_SUB_BACKUP_IMGSENSOR =
  - Build *kernel* driver and Android system. Then update boot.img and system.img

How to enable AF Feature

- To do list for enable AF feature
  - Put sensor driver within custom common part
    - File
      - alps\mediatek\custom\common\kernel\lens\${MODULE}
    - Sample
      - alps\mediatek\custom\common\kernel\lens\fm50af
  - Modify project make file
    - File
      - alps\mediatek\config\$project$\ProjectConfig.mk
    - Sample
      - alps\mediatek\config\mt6573_evb\ProjectConfig.mk
      - CUSTOM_HAL_LENS = fm50af
      - CUSTOM_KERNEL_LENS = fm50af
  - Build *kernel* driver and Android system. Then update boot.img and system.img

How to enable Flashlight Feature

- To do list for enable Flashlight feature
  - Put sensor driver within custom common part
    - File
      - alps\mediatek\custom\common\kernel\flashlight\${MODULE}
    - Sample
      - alps\mediatek\custom\common\kernel\flashlight\dummy_flashlight
  - Modify project make file
    - File
      - alps\mediatek\config\$project$\ProjectConfig.mk
    - Sample
      - alps\mediatek\config\mt6573_evb\ProjectConfig.mk
      - CUSTOM_HAL_FLASHLIGHT = dummy_flashlight
      - CUSTOM_KERNEL_FLASHLIGHT = dummy_flashlight
  - Build *kernel* driver and Android system. Then update boot.img and system.img

MT6516 Android Flashlight Support List

Flash light type	Description
dummy_flashlight	No support flash light
peak_flashlight	The flash only turn on some frames during image capture
constant_flashlight	The flash only turn on some frames during image capture
torch_flashlight	The flash always turn on during image capture

Type \ release	MT6516 YUV	MT6516 RAW	MT6573 YUV	MT6573 RAW
Peak	X	O	X	X
Torch	O	O	O	O
Constant	X	X	O	O

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

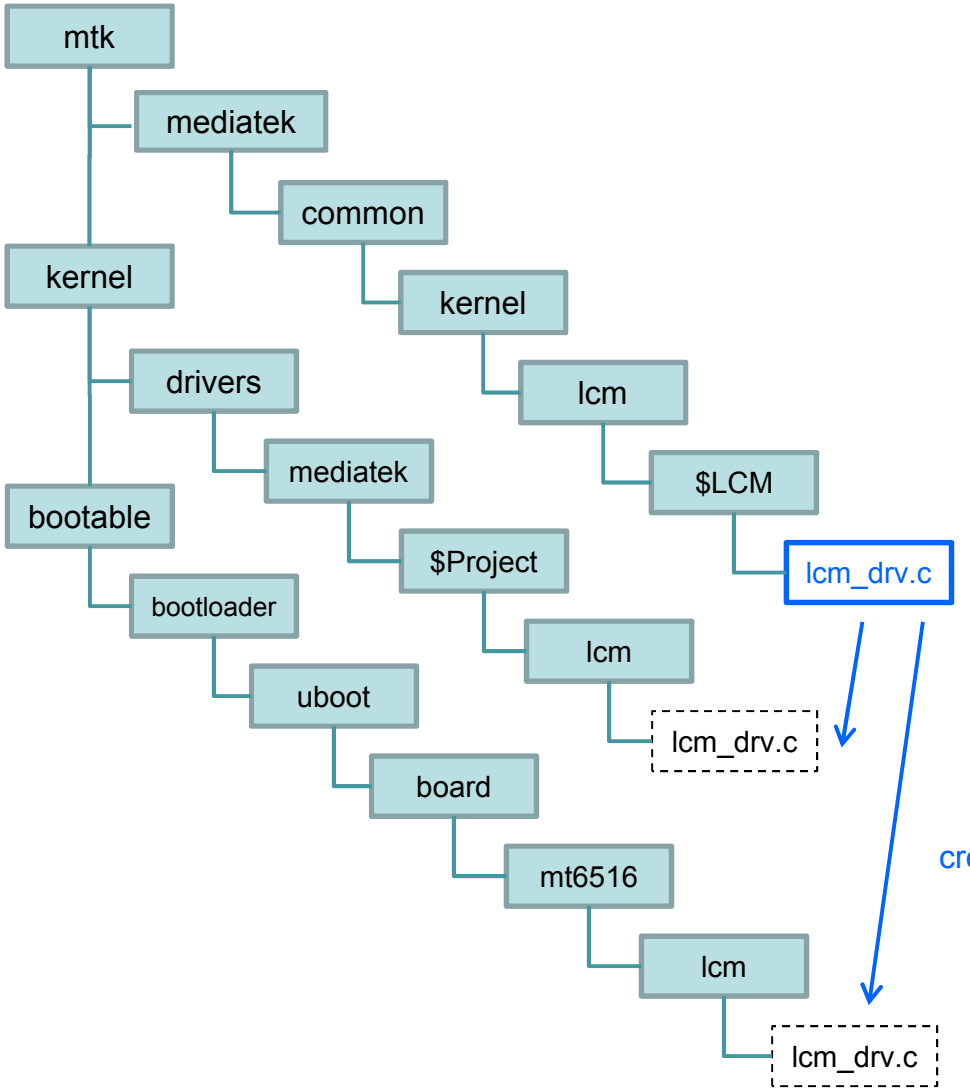
Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix

• OFN	Jogball	USB	FM
• SDHC	GPS	BT	
• Camera	***LCM Porting***	RTC	
• Factory	Recovery	Modem	

Source Tree



Uboot and Kernel share the same LCM driver source code

DBI Interface LCM Case Study

- LCM : SPFD5408A
- Interface : 16-bit 80 System Bus Interface
- LCD Size : 240 x 320

DBI Interface LCM Case Study

- Step 0: Make LCM driver folder and modify the project make file

Make LCM driver folder

```
$ mkdir ./mtk/mediatek/common/kernel/lcm/spfd5461a
```

Create lcm_drv.c

lcm_drv.c

```
const LCM_DRIVER* LCM_GetDriver()
{
    static const LCM_DRIVER LCM_DRV =
    {
        .set_util_funcs = lcm_set_util_funcs,
        .get_params     = lcm_get_params,
        .init           = lcm_init,
        .suspend        = lcm_suspend,
        .resume         = lcm_resume,
        .update         = lcm_update
    };

    return &LCM_DRV;
}
```

Modify the project make file

./mediatek/config/$Project$/ProjectConfig.mk

```
BRANCH=ALPS.10X
MTK_PLATFORM=MT6516
CHIP_VER=S01
...
CUSTOM_KERNEL_LCM=spfd5461a
...
```

/bootable/bootloader/uboot/XX_boot

UBOOT_LCM_CONFIG

DBI Interface LCM Case Study

- Step 1 : Fill LCM parameters

```
#define FRAME_WIDTH  (240)
#define FRAME_HEIGHT (320)
```

```
static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DBI;
    params->ctrl   = LCM_CTRL_PARALLEL_DBI;
    params->dbi.port = 1;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dbi.data_width                = LCM_DBI_DATA_WIDTH_16BITS;
    params->dbi.data_format.color_order   = LCM_COLOR_ORDER_RGB;
    params->dbi.data_format.trans_seq     = LCM_DBI_TRANS_SEQ_MSB_FIRST;
    params->dbi.data_format.padding       = LCM_DBI_PADDING_ON_LSB;
    params->dbi.data_format.format        = LCM_DBI_FORMAT_RGB565;
    params->dbi.data_format.width         = LCM_DBI_DATA_WIDTH_16BITS;

    params->dbi.clock_freq                = LCM_DBI_CLOCK_FREQ_52M;
    params->dbi.cpu_write_bits            = LCM_DBI_CPU_WRITE_16_BITS;
    params->dbi.parallel.write_setup      = 0;
    params->dbi.parallel.write_hold       = 3;
    params->dbi.parallel.write_wait       = 3;
    params->dbi.parallel.read_setup       = 2;
    params->dbi.parallel.read_latency     = 19;
    params->dbi.parallel.wait_period      = 0;

    params->dbi.io_driving_current        = LCM_DRIVING_CURRENT_8MA;
}
```

Configure them according to the HW connection and LCM feature

DBI Interface LCM Case Study

- Step 1 : Fill LCM parameters

```
#define FRAME_WIDTH  (240)
#define FRAME_HEIGHT (320)

static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DBI;
    params->ctrl   = LCM_CTRL_PARALLEL_DBI;
    params->dbi.port = 1;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dbi.data_width                = LCM_DBI_DATA_WIDTH_16BITS;
    params->dbi.data_format.color_order   = LCM_COLOR_ORDER_RGB;
    params->dbi.data_format.trans_seq     = LCM_DBI_TRANS_SEQ_MSB_FIRST;
    params->dbi.data_format.padding       = LCM_DBI_PADDING_ON_LSB;
    params->dbi.data_format.format        = LCM_DBI_FORMAT_RGB565;
    params->dbi.data_format.width         = LCM_DBI_DATA_WIDTH_16BITS;

    params->dbi.clock_freq                = LCM_DBI_CLOCK_FREQ_52M;
    params->dbi.cpu_write_bits            = LCM_DBI_CPU_WRITE_16_BITS;
    params->dbi.parallel.write_setup      = 0;
    params->dbi.parallel.write_hold       = 3;
    params->dbi.parallel.write_wait       = 3;
    params->dbi.parallel.read_setup       = 2;
    params->dbi.parallel.read_latency     = 19;
    params->dbi.parallel.wait_period      = 0;

    params->dbi.io_driving_current        = LCM_DRIVING_CURRENT_8MA;
}
```

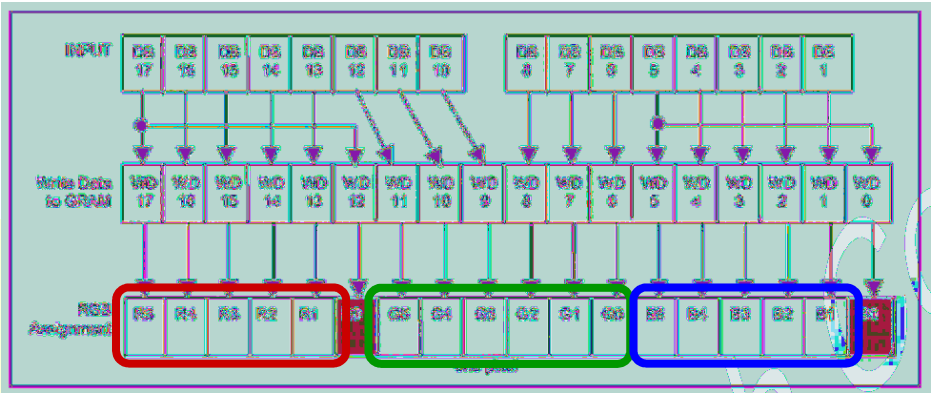


Figure 6-9 16-bit interface(65,536 colors) TRIREG= 0

Configure them according to the RGB data pin assignment of the LCM datasheet

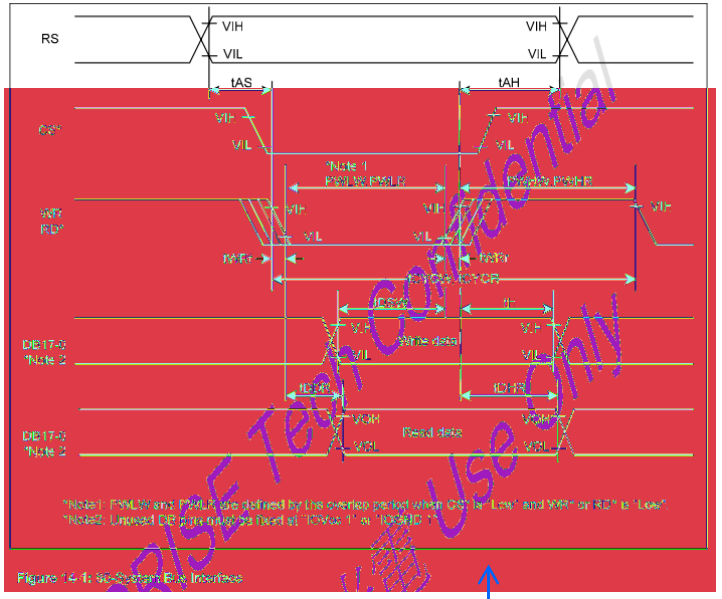
DBI Interface LCM Case Study

- Step 1 : Fill LCM parameters

Configure waveform timing according to the requirement specified in the LCM datasheet

```
#define FRAME_WIDTH  (240)
#define FRAME_HEIGHT (320)

static void lcm_get_params(LCM_PARAMS
{
    memset(params, 0, sizeof(LCM_PARA

    params->type   = LCM_TYPE_DBI;
    params->ctrl   = LCM_CTRL_PARALLEL_DBI;
    params->dbi.port = 1;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dbi.data_width                = LCM_DBI_DATA_WIDTH_16BITS;
    params->dbi.data_format.color_order   = LCM_COLOR_ORDER_RGB;
    params->dbi.data_format.trans_seq     = LCM_DBI_TRANS_SEQ_MSB_FIRST;
    params->dbi.data_format.padding       = LCM_DBI_PADDING_ON_LSB;
    params->dbi.data_format.format        = LCM_DBI_FORMAT_RGB565;
    params->dbi.data_format.width         = LCM_DBI_DATA_WIDTH_16BITS;

    params->dbi.clock_freq                = LCM_DBI_CLOCK_FREQ_52M;
    params->dbi.cpu_write_bits            = LCM_DBI_CPU_WRITE_16_BITS;
    params->dbi.parallel.write_setup      = 0;
    params->dbi.parallel.write_hold       = 3;
    params->dbi.parallel.write_wait       = 3;
    params->dbi.parallel.read_setup       = 2;
    params->dbi.parallel.read_latency     = 19;
    params->dbi.parallel.wait_period      = 0;

    params->dbi.io_driving_current        = LCM_DRIVING_CURRENT_8MA;
}
```

Item	Symbol	Unit	Min.	Typ.	Max.
Address Hold Time	tAH	ns	2	-	-
Write data setup time	tDSW	ns	25	-	-
Write data hold time	tH	ns	10	-	-
Read data delay time	tDDR	ns	-	-	150
Read data hold time	tDHR	ns	5	-	-



Confirm if the waveform can be supported by MT6516 DBI

DBI Interface LCM Case Study

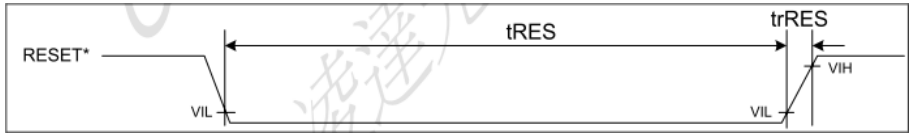
- Step 2 : Implement LCM init function

```
#define SET_RESET_PIN(v)    (lcm_util.set_reset_pin((v)))

#define UDELAY(n) (lcm_util.udelay(n))
#define MDELAY(n) (lcm_util.mdelay(n))
```

```
static __inline void send_ctrl_cmd(unsigned int cmd)
{
    lcm_util.send_cmd(cmd);
}

static __inline void send_data_cmd(unsigned int data)
{
    lcm_util.send_data(data);
}

static __inline void set_lcm_register(unsigned int regIndex,
                                      unsigned int regData)
{
    send_ctrl_cmd(regIndex);
    send_data_cmd(regData);
}
```

```
static void lcm_init(void)
{
    SET_RESET_PIN(0);
    MDELAY(2);
    SET_RESET_PIN(1);
    MDELAY(2);

    set_lcm_register(0x10, 0x10B0);   // SAP, BT[3:0], AP, DSTB, SLP, STB
    MDELAY(POWER_ON_SEQ_DELAY);
    set_lcm_register(0x11, 0x0007);   // DC1[2:0], DC0[2:0], VC[2:0]
    MDELAY(POWER_ON_SEQ_DELAY);
    set_lcm_register(0x17, 0x0001);
    ...
}
```

Reset LCM and delay for a while according to the timing requirement specified in LCM datasheet



Item	Symbol	Unit	Min.	Typ.	Max.
Reset low-level width	tRES	ms	1	–	–
Reset rise time	trRES	µs	–	–	10

LCM init code provided by vendor

DBI Interface LCM Case Study

- Step 3 : Implement LCM update function

```
static void lcm_update(unsigned int x, unsigned int y,
                       unsigned int width, unsigned int height)
{
    unsigned int x0 = x;
    unsigned int y0 = y;
    unsigned int x1 = x0 + width - 1;
    unsigned int y1 = y0 + height - 1;

    set_lcm_register(0x50, x0);
    set_lcm_register(0x51, x1);
    set_lcm_register(0x52, y0);
    set_lcm_register(0x53, y1);
    set_lcm_register(0x20, x0);
    set_lcm_register(0x21, y0);

    send_ctrl_cmd(0x22);
}
```

Send the block update commands to LCM

Window Horizontal RAM Address Start (R50h)
Window Horizontal RAM Address End (R51h)
Window Vertical RAM Address Start (R52h)
Window Vertical RAM Address End (R53h)
GRAM Address Set (Horizontal Address) (R20h)
GRAM Address Set (Vertical Address) (R21h)

Write Data to GRAM (R22h)

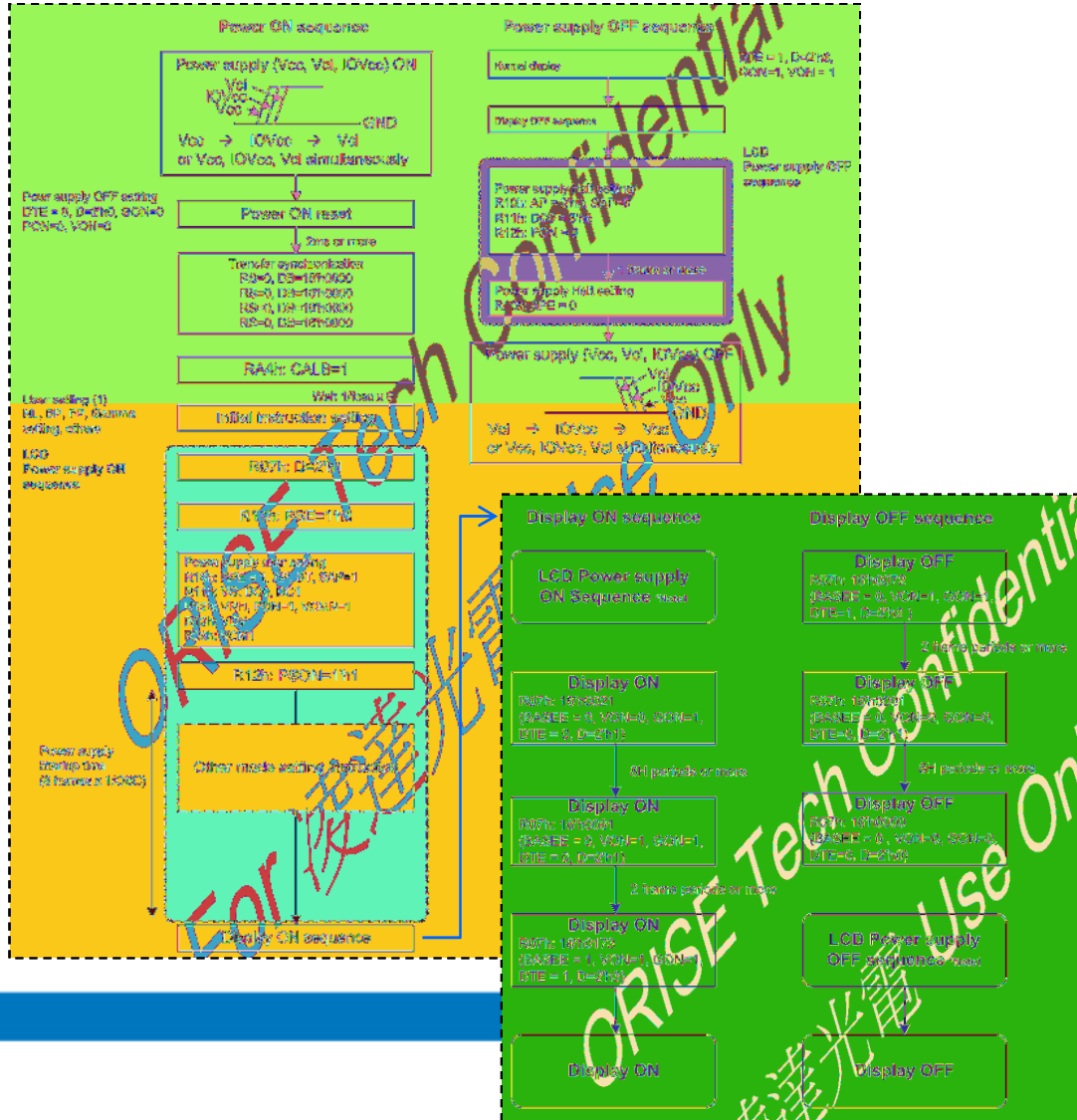
DBI Interface LCM Case Study

- Step 4 : Implement LCM suspend / resume function

```
static void lcm_suspend(void)
{
    set_lcm_register(0x07, 0x0101);
    MDELAY(10);
    set_lcm_register(0x07, 0x0000);    // display off
    MDELAY(10);
    set_lcm_register(0x10, 0x10B2);
    MDELAY(50);
}
```

```
static void lcm_resume(void)
{
    set_lcm_register(0x10, 0x10B0);
    MDELAY(25);
    set_lcm_register(0x07, 0x0173);
    MDELAY(175);
}
```

Send suspend / resume commands to LCM



DPI Interface LCM Case Study

- LCM : LG4571
- Interface
  - 24-bit RGB parallel data interface
  - 3-wire SPI command interface
- LCD Size : 480 x 800

DPI Interface LCM Case Study

- Step 0: Make LCM driver folder and modify the project make file

Make LCM driver folder

```
$ mkdir ./mediatek/custom/common/kernel/lcm/lg4571
```

Create lcm_drv.c

lcm_drv.c

```
const LCM_DRIVER* LCM_GetDriver()
{
    static const LCM_DRIVER LCM_DRV =
    {
        .set_util_funcs = lcm_set_util_funcs,
        .get_params     = lcm_get_params,
        .init           = lcm_init,
        .suspend        = lcm_suspend,
        .resume         = lcm_resume,
        .update         = lcm_update
    };

    return &LCM_DRV;
}
```

DPI interface LCM need not implement lcm_update() callback

Modify the project make file

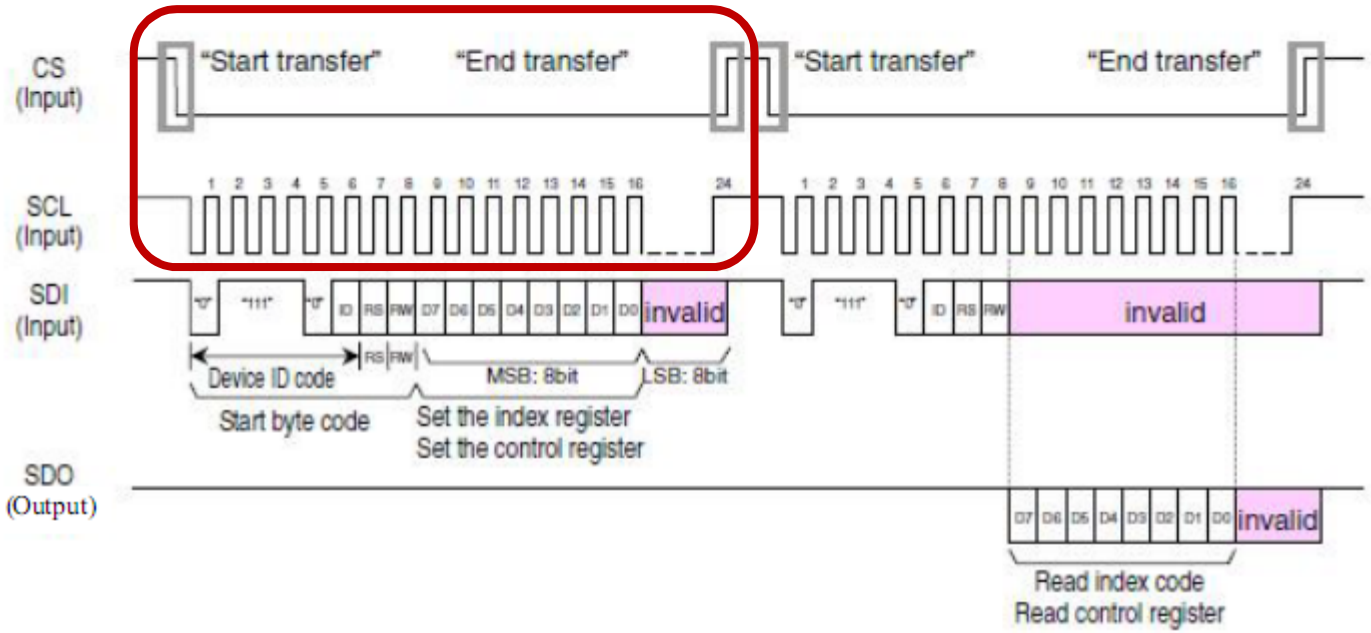
./mtk/make/$(project).mak

```
BRANCH=ALPS.10X
MTK_PLATFORM=MT6516
CHIP_VER=S01
...
CUSTOM_KERNEL_LCM=lg4571
...
```

DPI Interface LCM Case Study

- Step 1: Decide use which I/F to configure the LCM internal registers

Keep CS low during 24 bit data transaction → Serial DBI can't fit the CS waveform requirement, use GPIO I/F instead



DPI Interface LCM Case Study

- Step 2 : SW simulate the waveform via GPIO

```
#include <mach/mt6516_gpio.h>
```

defined in the customized GPIO header file

```
#define LSAO_GPIO_PIN  (GPIO_DISP_LSAO_PIN)
#define LSCE_GPIO_PIN  (GPIO_DISP_LSCE_PIN)
#define LSCK_GPIO_PIN  (GPIO_DISP_LSCK_PIN)
#define LSDA_GPIO_PIN  (GPIO_DISP_LSDA_PIN)
```

```
#define SET_LSCE_LOW    SET_GPIO_OUT(LSCE_GPIO_PIN, 0)
#define SET_LSCE_HIGH   SET_GPIO_OUT(LSCE_GPIO_PIN, 1)
#define SET_LSCK_LOW    SET_GPIO_OUT(LSCK_GPIO_PIN, 0)
#define SET_LSCK_HIGH   SET_GPIO_OUT(LSCK_GPIO_PIN, 1)
#define SET_LSDA_LOW    SET_GPIO_OUT(LSDA_GPIO_PIN, 0)
#define SET_LSDA_HIGH   SET_GPIO_OUT(LSDA_GPIO_PIN, 1)
```

```
static __inline void spi_send_data(unsigned int data)
{
    unsigned int i;

    SET_LSCE_LOW;
    UDELAY(1);
    SET_LSCK_HIGH;
    SET_LSDA_HIGH;
    UDELAY(1);

    for (i = 0; i < 24; ++ i)
    {
        SET_LSCK_LOW;
        if (data & (1 << 23)) {
            SET_LSDA_HIGH;
        } else {
            SET_LSDA_LOW;
        }
        UDELAY(1);
        SET_LSCK_HIGH;
        UDELAY(1);
        data <<= 1;
    }

    SET_LSDA_HIGH;
    SET_LSCE_HIGH;
}
```

DPI Interface LCM Case Study

- Step 3 : Fill DPI LCM parameters – control mode

```
#define FRAME_WIDTH  (480)
#define FRAME_HEIGHT (800)
```

```
static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DPI;
    params->ctrl   = LCM_CTRL_GPIO;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dpi.mipi_pll_clk_ref  = 0;
    params->dpi.mipi_pll_clk_div1 = 42;
    params->dpi.mipi_pll_clk_div2 = 10;
    params->dpi.dpi_clk_div       = 2;
    params->dpi.dpi_clk_duty      = 1;

    params->dpi.clk_pol           = LCM_POLARITY_FALLING;
    params->dpi.de_pol            = LCM_POLARITY_FALLING;
    params->dpi.vsync_pol         = LCM_POLARITY_FALLING;
    params->dpi.hsync_pol         = LCM_POLARITY_FALLING;

    params->dpi.hsync_pulse_width = 4;
    params->dpi.hsync_back_porch  = 10;
    params->dpi.hsync_front_porch = 16;
    params->dpi.vsync_pulse_width = 2;
    params->dpi.vsync_back_porch  = 2;
    params->dpi.vsync_front_porch = 14;

    params->dpi.format            = LCM_DPI_FORMAT_RGB888;
    params->dpi.rgb_order         = LCM_COLOR_ORDER_RGB;
    params->dpi.is_serial_output  = 0;

    params->dpi.intermediat_buffer_num = 2;

    params->dpi.io_driving_current = LCM_DRIVING_CURRENT_2MA;
}
```

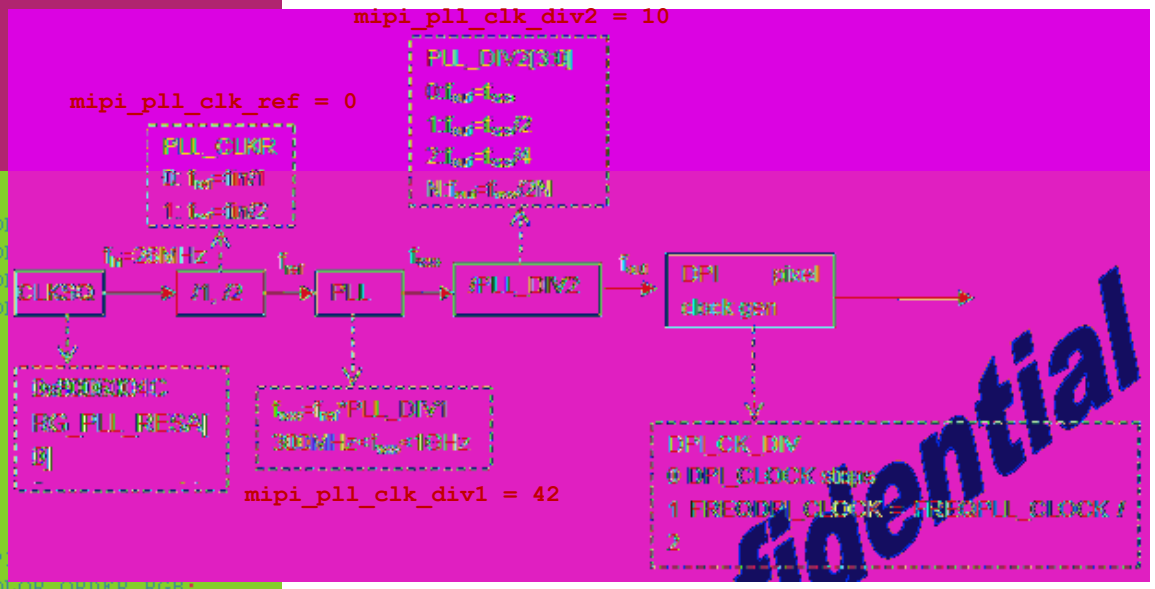
Control LCM registers via GPIO

DPI Interface LCM Case Study

- Step 3 : Fill DPI LCM parameters – clock frequency

```
static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DPI;
    params->ctrl   = LCM_CTRL_GPIO;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dpi.mipi_pll_clk_ref  = 0;
    params->dpi.mipi_pll_clk_div1 = 42;
    params->dpi.mipi_pll_clk_div2 = 10;
    params->dpi.dpi_clk_div       = 2;
    params->dpi.dpi_clk_duty      = 1;

    params->dpi.clk_pol           = LCM_POL
    params->dpi.de_pol            = LCM_POL
    params->dpi.vsync_pol         = LCM_POL
    params->dpi.hsync_pol         = LCM_POL

    params->dpi.hsync_pulse_width = 4;
    params->dpi.hsync_back_porch  = 10;
    params->dpi.hsync_front_porch = 16;
    params->dpi.vsync_pulse_width = 2;
    params->dpi.vsync_back_porch  = 2;
    params->dpi.vsync_front_porch = 14;

    params->dpi.format            = LCM_DP
    params->dpi.rgb_order         = LCM_COLOR_ORDER_RGB;
    params->dpi.is_serial_output  = 0;

    params->dpi.intermediat_buffer_num = 2;

    params->dpi.io_driving_current = LCM_DRIVING_CURRENT_2MA;
}
```

Item	min	typ	max	Unit
fFLM	(60)	65	(70)	Hz
PCLK	(25.13)	27.22	(29.32)	MHz

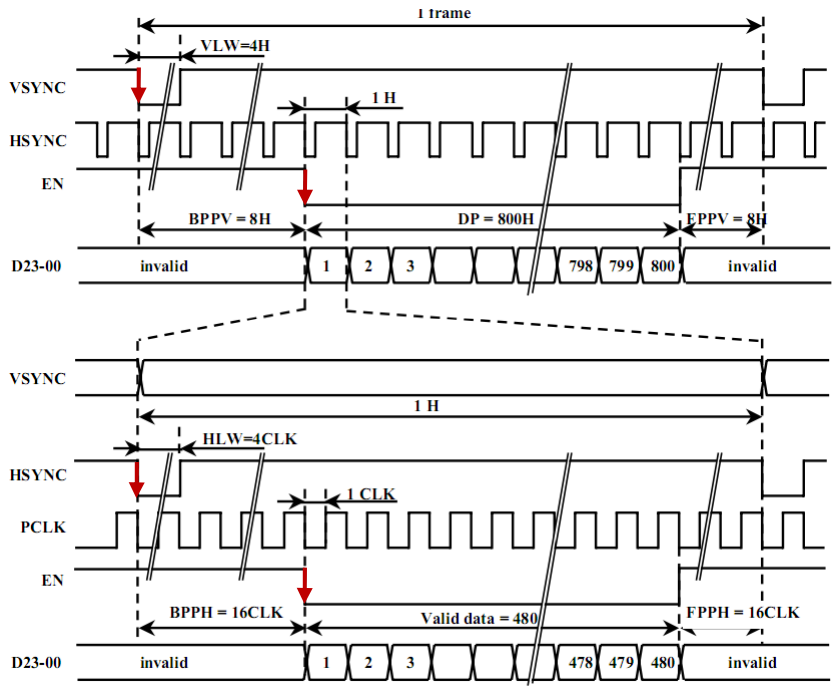


PCLK = (((26MHz / 1) * 42) / 20) / 2 = 27.3MHz

DPI Interface LCM Case Study

- Step 3 : Fill DPI LCM parameters – polarity

```
static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DPI;
    params->ctrl   = LCM_CTRL_GPIO;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dpi.mipi_pll_clk_ref  = 0;
    params->dpi.mipi_pll_clk_div1 = 42;
    params->dpi.mipi_pll_clk_div2 = 10;
    params->dpi.dpi_clk_div       = 2;
    params->dpi.dpi_clk_duty      = 1;

    params->dpi.clk_pol           = LCM_POLARITY_FALLING;
    params->dpi.de_pol            = LCM_POLARITY_FALLING;
    params->dpi.vsync_pol         = LCM_POLARITY_FALLING;
    params->dpi.hsync_pol         = LCM_POLARITY_FALLING;

    params->dpi.hsync_pulse_width = 4;
    params->dpi.hsync_back_porch  = 10;
    params->dpi.hsync_front_porch = 18;
    params->dpi.vsync_pulse_width = 2;
    params->dpi.vsync_back_porch  = 2;
    params->dpi.vsync_front_porch = 14;

    params->dpi.format            = LCM_DPI_FORMAT_RGB888;
    params->dpi.rgb_order         = LCM_COLOR_ORDER_RGB;
    params->dpi.is_serial_output  = 0;

    params->dpi.intermediat_buffer_num = 2;

    params->dpi.io_driving_current = LCM_DRIVING_CURRENT_2MA;
}
```

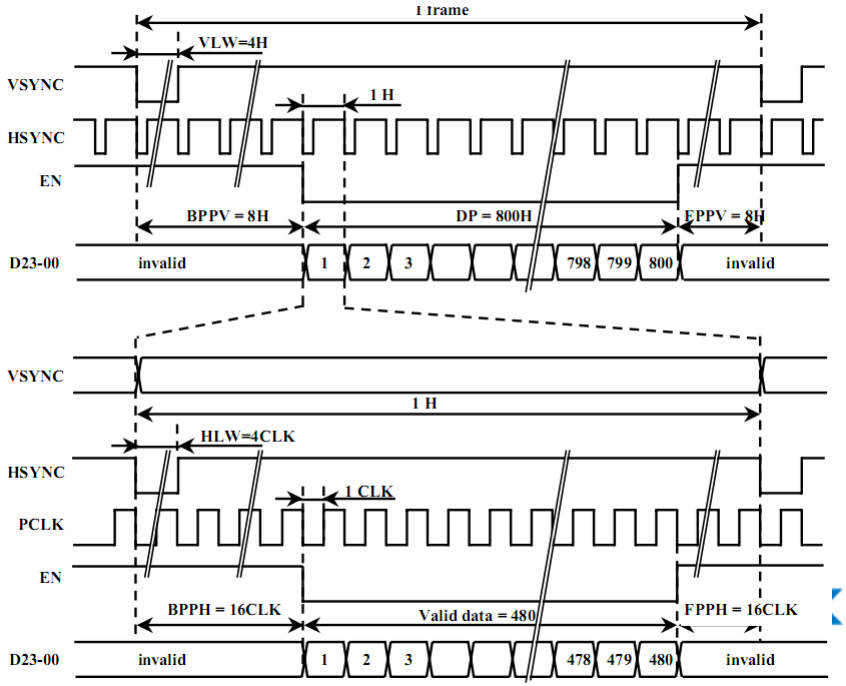


DPI Interface LCM Case Study

- Step 3 : Fill DPI LCM parameters – blanking timing

```
static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DPI;
    params->ctrl   = LCM_CTRL_GPIO;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dpi.mipi_pll_clk_ref  = 0;
    params->dpi.mipi_pll_clk_div1 = 42;
    params->dpi.mipi_pll_clk_div2 = 10;
    params->dpi.dpi_clk_div       = 2;
    params->dpi.dpi_clk_duty      = 4;

    params->dpi.clk_pol           = LCM_POLARITY_FALLING;
    params->dpi.de_pol            = LCM_POLARITY_FALLING;
    params->dpi.vsync_pol         = LCM_POLARITY_FALLING;
    params->dpi.hsync_pol         = LCM_POLARITY_FALLING;

    params->dpi.hsync_pulse_width = 4;
    params->dpi.hsync_back_porch  = 10;
    params->dpi.hsync_front_porch = 18;
    params->dpi.vsync_pulse_width = 2;
    params->dpi.vsync_back_porch  = 2;
    params->dpi.vsync_front_porch = 14;

    params->dpi.format            = LCM_DPI_FORMAT_RGB888;
    params->dpi.rgb_order         = LCM_COLOR_ORDER_RGB;
    params->dpi.is_serial_output  = 0;

    params->dpi.intermediat_buffer_num = 2;

    params->dpi.io_driving_current = LCM_DRIVING_CURRENT_2MA;
}
```

Item	min	typ	max	Unit
HLW	1	4	4	CLK
BPPH	10	14	68	CLK
FPPH	18	18	100	CLK
VLW	1	4	4	H
BPPV	3	4	4	H
FPPV	14	14	14	H
DP	-	800	-	H



DPI Interface LCM Case Study

- Step 3 : Fill DPI LCM parameters – output format

```
static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DPI;
    params->ctrl   = LCM_CTRL_GPIO;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;

    params->dpi.mipi_pll_clk_ref  = 0;
    params->dpi.mipi_pll_clk_div1 = 42;
    params->dpi.mipi_pll_clk_div2 = 10;
    params->dpi.dpi_clk_div       = 2;
    params->dpi.dpi_clk_duty      = 1;

    params->dpi.clk_pol           = LCM_POLARITY_FALLING;
    params->dpi.de_pol            = LCM_POLARITY_FALLING;
    params->dpi.vsync_pol         = LCM_POLARITY_FALLING;
    params->dpi.hsync_pol         = LCM_POLARITY_FALLING;

    params->dpi.hsync_pulse_width = 4;
    params->dpi.hsync_back_porch  = 10;
    params->dpi.hsync_front_porch = 18;
    params->dpi.vsync_pulse_width = 2;
    params->dpi.vsync_back_porch  = 2;
    params->dpi.vsync_front_porch = 14;

    params->dpi.format            = LCM_DPI_FORMAT_RGB888;
    params->dpi.rgb_order         = LCM_COLOR_ORDER_RGB;
    params->dpi.is_serial_output  = 0;

    params->dpi.intermediat_buffer_num = 2;

    params->dpi.io_driving_current = LCM_DRIVING_CURRENT_2MA;
}
```

24-bit RGB Interface mode :

RGB Interface Input data	D 23	D 22	D 21	D 20	D 19	D 18	D 17	D 16
Instruction Register settings SS	Instruction Register settings BGR	RGB Equiv.	R7	R6	R5	R4	R3	R2	R1	R0
0	0	Output pin	S(3n + 1)							
0	1		S(3n + 3)							
1	0		S(1438 – 3n)							
1	1		S(1440 – 3n)							

RGB Interface Input data	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8
Instruction Register settings SS	Instruction Register settings BGR	RGB equiv.	G7	G6	G5	G4	G3	G2	G1	G0
0	0	Output pin	S(3n + 2)							
0	1		S(3n + 2)							
1	0		S(1439 – 3n)							
1	1		S(1439 – 3n)							

RGB Interface Input data	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
Instruction Register settings SS	Instruction Register settings BGR	RGB equiv.	B7	B6	B5	B4	B3	B2	B1	B0
0	0	Output pin	S(3n + 3)							
0	1		S(3n + 1)							
1	0		S(1440 – 3n)							
1	1		S(1438 – 3n)							

DPI Interface LCM Case Study

- Step 4 : Implement LCM init function
- Step 5 : Implement LCM suspend / resume function

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix
- OFN
- SDHC
- Camera
- Factory
- Jogball
- GPS
- LCM Porting
- Recovery
- USB
- BT
- ***RTC***
- Modem
- FM

RTC in Preloader

- When initializing RTC HW, we will initialize RTC time counters by using a set of time

File Name	Location
Cust_rtc.h	Android 2.2 alps\mtk\src\custom\[project]\preloader\custom\inc Android 2.3 Alps\mediatek\custom\[project]\preloader\inc

```
/*
 * Default values for RTC initialization
 * Year (YEA)        : 1970 ~ 2037
 * Month (MTH)       : 1 ~ 12
 * Day of Month (DOM): 1 ~ 31
 * Day of Week (DOW) : 0 (Sun.) ~ 6 (Sat.)
 */
#define RTC_DEFAULT_YEA     2010
#define RTC_DEFAULT_MTH     1
#define RTC_DEFAULT_DOM     1
```

RTC_DEFAULT_YEA	Year	1970 ~ 2037
RTC_DEFAULT_MTH	Month	1 ~ 12
RTC_DEFAULT_DOM	Day of Month	1 ~ 31

RTC in Kernel

- The time in 32-bit Linux will overflow at 2038/01/19 03:14:07 but RTC time counters still keep running
  - If the user does not reset the system time, there may be abnormal exceptions occurred, especially after rebooting the system
- You can enable RTC_OVER_TIME_RESET which will reset RTC time counters when Kernel reads RTC time and RTC time is over 2038/01/19 03:14:07
  - If the user reboots the system, the system time will be the normal state, not overflow state

RTC in Kernel

- RTC_OVER_TIME_RESET (default: Yes)

File Name	Location
Rtc-mt6516.c Rtc-mt6573.c	Android 2.2 alps\mtk\src\custom\[project]\kernel\rtc\rtc Android 2.3 Alps\mediatek\custom\[project]\kernel\rtc\rtc

```
/*
 * Reset to default date if RTC time is over 2038/1/19 3:14:7
 * Year (YEA)         : 1970 ~ 2037
 * Month (MTH)        : 1 ~ 12
 * Day of Month (DOM): 1 ~ 31
 * Day of Week (DOW) : 0 (Sun.) ~ 6 (Sat.)
 */
#define RTC_OVER_TIME_RESET RTC_YES
#define RTC_DEFAULT_YEA     2010
#define RTC_DEFAULT_MTH     1
#define RTC_DEFAULT_DOM     1
```

RTC_DEFAULT_YEA	Year	1970 ~ 2037
RTC_DEFAULT_MTH	Month	1 ~ 12
RTC_DEFAULT_DOM	Day of Month	1 ~ 31

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

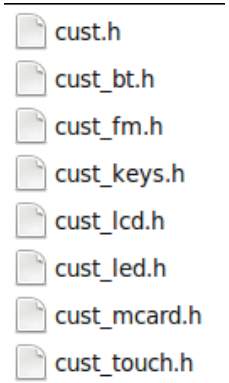
Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix

• OFN	Jogball	USB	FM
• SDHC	GPS	BT	
• Camera	LCM Porting	RTC	
• ***Factory***	Recovery	Modem	

Key mapping

- KEY define
  - Use Power key + Factory key to enter factory mode
    - Location: alps\mediatek\custom\${project_name}\uboot\inc\configs\mt6573_xx.h
    - #define MT65XX_FACTORY_KEY {key number}
  - Customization Files
    - Location: alps\mediatek\custom\${project_name}*factory**inc*\



Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

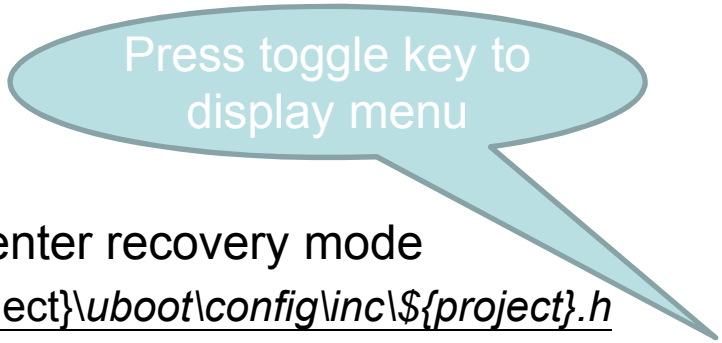
Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix
- OFN
- SDHC
- Camera
- Factory
- Jogball
- GPS
- LCM Porting
- ***Recovery***
- USB
- BT
- RTC
- Modem
- FM

Key mapping



- KEY define
  - Use Power key + Recovery keyt to enter recovery mode
    - Location: alps\mtk\src\custom\${project}*uboot\config\inc\${project}.h*
    - #define MT6516_RECOVERY_KEY {key number}
  - Change the toggle display key in the function
    - Location: alps\bootable\recovery\default_recovery_ui.c

```
int device_toggle_display(volatile char* key_pressed, int key_code) {
    return key_code == KEY_HOME;
}
```

  - Define the keys in Recovery Mode
    - Location: alps\mtk\src\custom\{project}\recovery\inc\cust_keys.h

```
#define RECOVERY_KEY_DOWN      KEY_DOWN
#define RECOVERY_KEY_VOLDOWN   KEY_VOLUMEDOWN
#define RECOVERY_KEY_UP        KEY_UP
#define RECOVERY_KEY_VOLUP     KEY_VOLUMEUP
#define RECOVERY_KEY_CENTER    KEY_OK
#define RECOVERY_KEY_RIGHT     KEY_BACK
#define RECOVERY_KEY_LEFT      KEY_CALL
```

Recovery Process

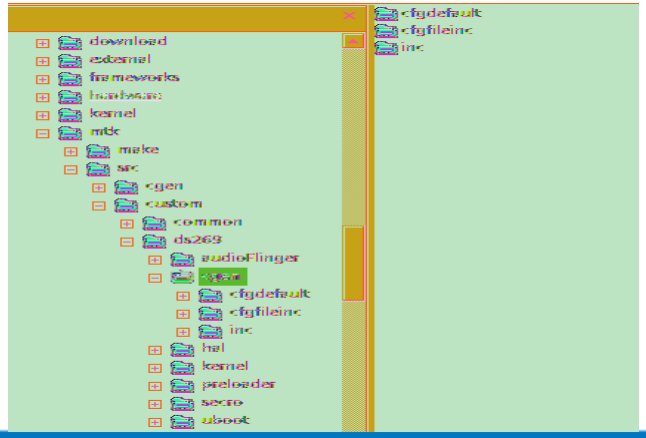
- **Build update image step by step**
  - ./makeMtk <project> otapackage
  - Copy out /target/product/<project> /< project >-ota-<mode>.<user_id>.zip to root of the directory of SD card and rename it to update.zip
- **Upgrade Procedure**
  - Press volume up and powerkey to enter recovery mode
  - Press home key to enter main menu
  - Use volume down key to select apply sdcard:update.zip
  - Press menu key to execute to upgrade procedure

Customization in NvRam

- For the different requirements of projects, NvRam modules also need to provide the supports of customization configurations, including default value and record data structure of NvRam files.
- There are two parts of NvRAM data
  - Common
    - For MTK platform NvRAM used
    - Customer can see the definition of related NVRAM record structure
    - But should not modify them
  - Customized for different projects
    - For customer NvRAM used
    - Customer can see the definition of related NVRAM record structure
    - Can modify them according to the requirements

Customization in NvRam

- The folder of NvRam customization is located in the path
  - *mtk\src\custom\[$PROJECT]\cgen*
- There are three folders in this customization folder
  - Cfgdefault
    - Used to define the default value of NvRam files
  - Cfgfileinc
    - Used to define the record data structure of NvRam file
  - Inc
    - Used to support general NvRam module functionalities



Customization in NvRam

- Should modify the file
  - *mtk\src\custom\[$PROJECT]\cgen\inc\CFG_file_info_custom.h*
  - Data structure of *g_akCFG_File_Custom*
- The information of NvRAM file
  - File path
    - The file path that the NvRAM files should be store
  - File version
  - Record size
  - Record numbers
  - The type of the default value
  - The default value

Customization in NvRam

- The data structure of *g_akCFG_File_Custom*

```
const TCFG_FILE g_akCFG_File_Custom[]=
{
        { "/nvram/APCFG/APRDCL/Audio_Sph",      VER(AP_CFG_RDCL_FILE_AUDIO_LID),      CFG_FILE_AUDIO_REC_SIZE,
          CFG_FILE_AUDIO_REC_TOTAL,             SIGNLE_DEFUALT_REC,                   (char *)&audio_custom_default},

        { "/nvram/APCFG/APRDEB/GPS",            VER(AP_CFG_CUSTOM_FILE_GPS_LID),      CFG_FILE_GPS_CONFIG_SIZE,
          CFG_FILE_GPS_CONFIG_TOTAL,            SIGNLE_DEFUALT_REC,                   (char *)&stGPSConfigDefault},
};
```

- The default value of *stGPSConfigDefault*

```
ap_nvram_gps_config_struct stGPSConfigDefault =
{
    /* "/dev/ttyMT1" */
    {'/','d','e','v','/','t','t','y','M','T','1',0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0},
    /* 0:s/w, 1:none, 2:h/w */
    1,

    /* 16.368MHz */
    16368000,
    /* 500ppb */
    500,
    /* 0:16.368MHz TCXO */
    0,

    /* 0:mixer-in, 1:internal-LNA */
    0,

    /* 0:none */
    0
};
```

Reset to Default

Type	Descriptions
SINGLE_DEFAULT_REC	If multiple records have same default value, this type should be used to minimize the Ram size. It only need define the default value of one record, NvRam module will use the default value of this record to initialize all of records
MULTIPLE_DEFAULT_REC	If NvRam has different default value for different records, this type should be used. It will use default value which is define in the cfg_file, then writes to NvRam file
DEFAULT_ZERO	The default value is 0, the property of default value will not be cared
DEFAULT_FF	The default value is 0xff, the property of default value will not be cared

Step by Step to Add NvRAM Data

1. Add one header file which describes the definition of its record data structure, record size and record numbers
   - In the path of *mtk\src\custom\[$PROJECT]\cgen\cfgfileinc*

```
#ifndef _CFG_CUSTOM1_FILE_H
#define _CFG_CUSTOM1_FILE_H

typedef struct
{
    unsigned int Array[1];
}File_Custom1_Struct;

#define CFG_FILE_CUSTOM1_REC_SIZE    sizeof(File_Custom1_Struct)
#define CFG_FILE_CUSTOM1_REC_TOTAL   1

#endif
```

2. Add header file which define its default value of NvRam file
   - In the path of *mtk\src\custom\[$PROJECT]\cgen\cfgdefault*

```
#ifndef _CFG_CUSTOM1_D_H
#define _CFG_CUSTOM1_D_H

File_Custom1_Struct stCustom1Default =
{
    1
};

#endif
```

Step by Step to Add NvRAM Data

3. Add one lid in the enum defination of "*CUSTOM_CFG_FILE_LID*" and define the version number of NvRam file
   - In the path of *mtk\src\custom\[$PROJECT]\cgen\inc\Custom_NvRam_LID.h*

```
/* the definition of file LID */
typedef enum
{
    AP_CFG_RDCL_FILE_AUDIO_LID=AP_CFG_CUSTOM_BEGIN_LID, //AP_CFG_CUSTOM_BEGIN_LID: this lid must not be changed, it is reserved for system.
    AP_CFG_CUSTOM_FILE_GPS_LID,
    AP_CFG_RDCL_FILE_META_LID,
    AP_CFG_CUSTOM_FILE_CUSTOM1_LID,
    AP_CFG_CUSTOM_FILE_CUSTOM2_LID,

    AP_CFG_CUSTOM_FILE_MAX_LID,
} CUSTOM_CFG_FILE_LID;


/* verno of data items */
/* audio file version */
#define AP_CFG_RDCL_FILE_AUDIO_LID_VERNO            "001"
/* META log and com port config file version */
#define AP_CFG_RDCL_FILE_META_LID_VERNO             "000"

/* custom2 file version */
#define AP_CFG_CUSTOM_FILE_CUSTOM1_LID_VERNO            "000"
/* custom2 file version */
#define AP_CFG_CUSTOM_FILE_CUSTOM2_LID_VERNO            "000"
/* GPS file version */
#define AP_CFG_CUSTOM_FILE_GPS_LID_VERNO            "000"
```

Step by Step to Add NvRAM Data

4. Add one include path which added in the step 1
   - In the path of *mtk\src\custom\[$PROJECT]\cgen\inc\custom_cfg_module_file.h*
5. Add one include path which added in the step 2
   - In the path of *mtk\src\custom\#PROJECT_NAME#\cgen\inc\custom_cfg_module_default.h*
6. Add the related information of NvRam file into the definition of "*g_akCFG_File_Custom*"
   - In the path of *mtk\src\custom\[$PROJECT]\cgen\inc\CFG_file_info_custom.h*
7. Add its related information, including record structure, NvRam lid, and record number
   - *In the path of mtk\src\custom\[$PROJECT]\cgen\inc\Custom_NvRam_data_item.h*

Outline

Overview
- Chip
- Build Command

Boot up
- Bootloader

Lights System

Battery Manager
- Battery Service
- Battery Charging
- Power Off Charging

Input/Output
- Keypad
- Touch
- Display

Connecting
- WIFI

Sensor System
- Sensor HAL
- G Sensor
- M Sensor
- ALS/PS Sensor

Audio

Misc.
- GPIO
- I2C
- Headset
- PMIC

Appendix

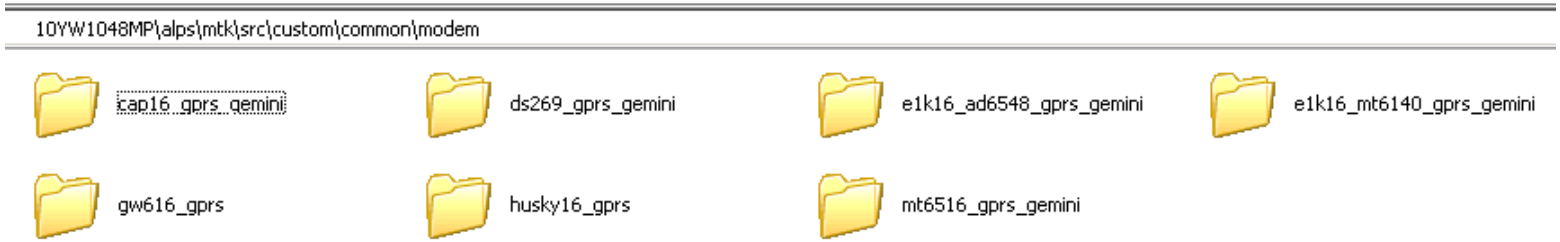
• OFN	Jogball	USB	FM
• SDHC	GPS	BT	
• Camera	LCM Porting	RTC	
• Factory	Recovery	***Modem***	

Modem Customization(1/2)

- Makefile Option (alps/mtk/make/[project_name].mak)

```
CUSTOM_COMMON_MODEM = E1K16_AD6548_GPRS_GEMINI
```

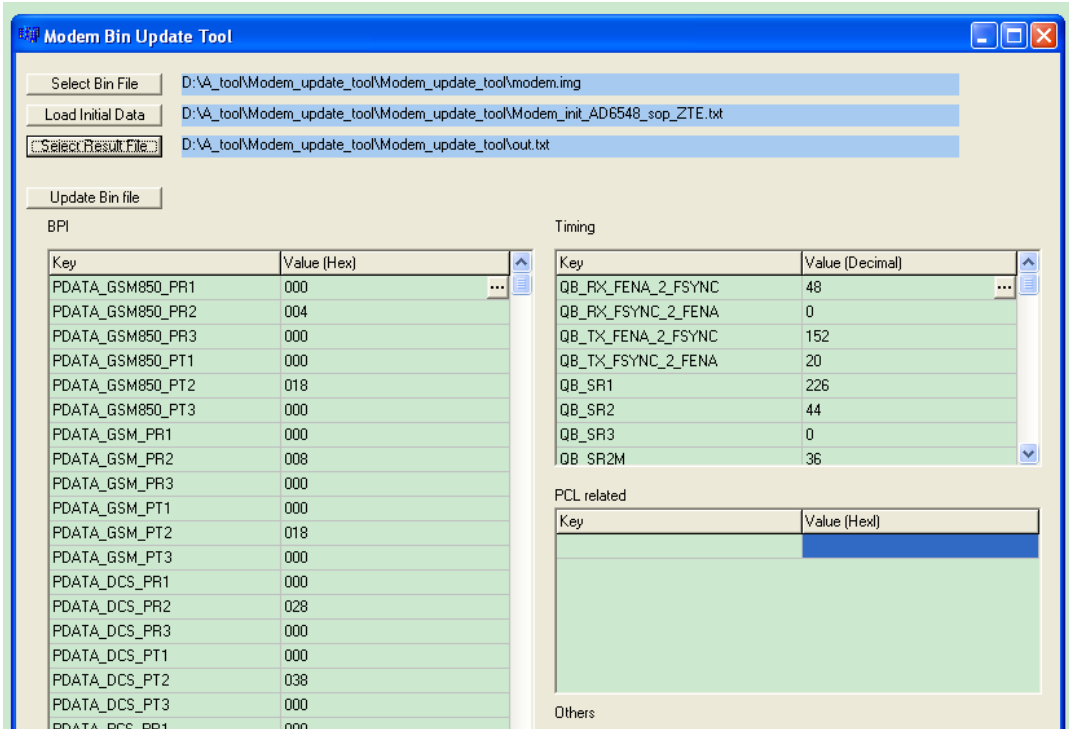
- The modem image will locate in path: alps/mtk/src/custom/common/modem



Modem Customization(2/2)

- Use Modem Bin Update Tool to Customization

File	Description
alps/mtk/src/custom/common/modem/${CUSTOM_COMMON_MODEM}	
Modem.img	The customization image file



MEDIATEK

End.