紫光展锐
UNISOC

介绍

WWW.UNISOC.COM

紫 光 展 锐 科 技

修改历史

版本号	日期	注释
V0.1	2020/04/24	初稿
V0.2	2020/06/05	更新模板
V0.3	2020/06/18	修订添加运行级别、内存分布等
V0.4	2020/07/06	修订添加启动模式描述

关键字

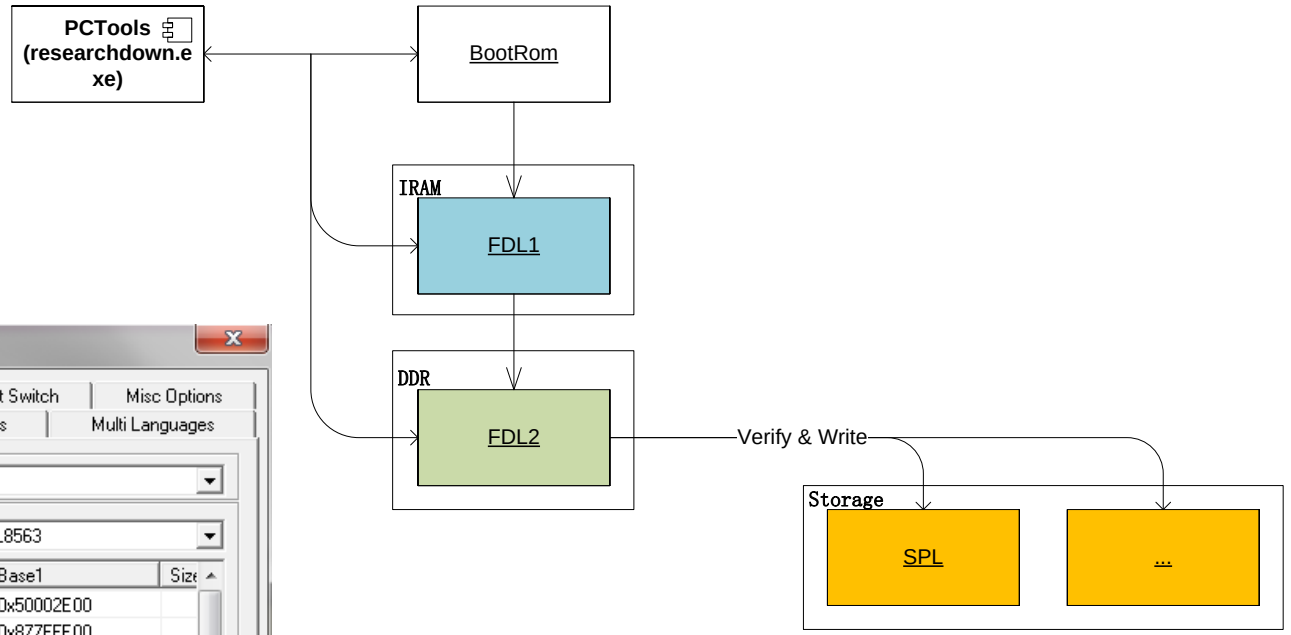
关键字 ：UBoot，U-Boot；

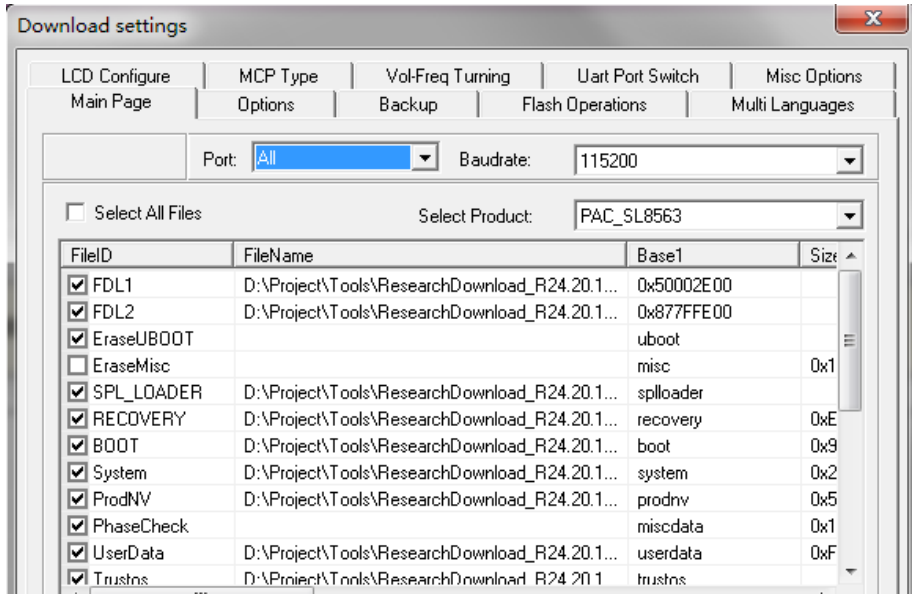
目录

基本流程

下载流程

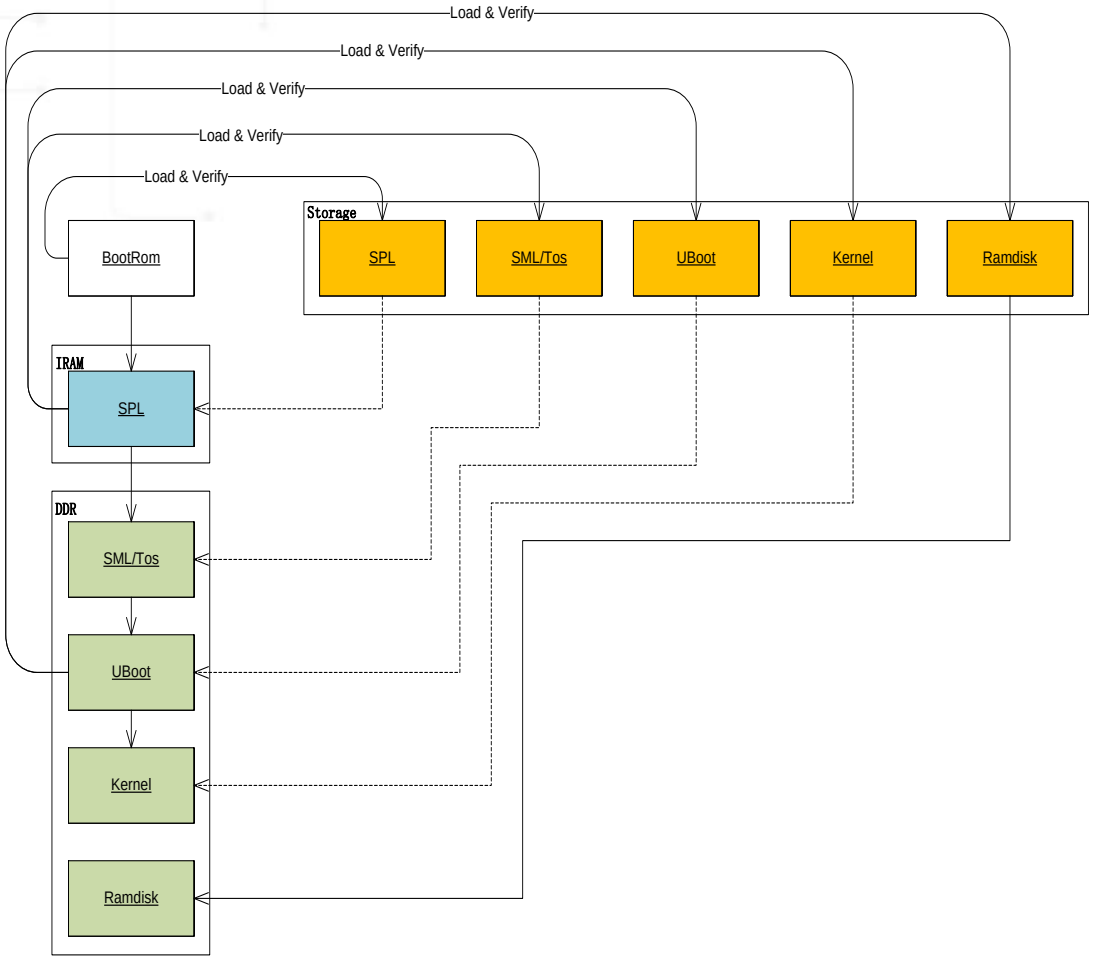
- BootRom 与下载工具交互建立通路，并下载FDL1到IRAM运行
- FDL1初始化DDR，并与工具交互，将FDL2下载到DDR，并运行
- FDL2与工具交互，下载其它image到Flash





下载流程

- 用BootRom 校验SPL 的Image 安全性并加载启动
- 用SPL 校验U-Boot 的安全性并加载启动
- 用U-Boot 校验Linux 的安全性并加载启动



02

运行级别

运行级别

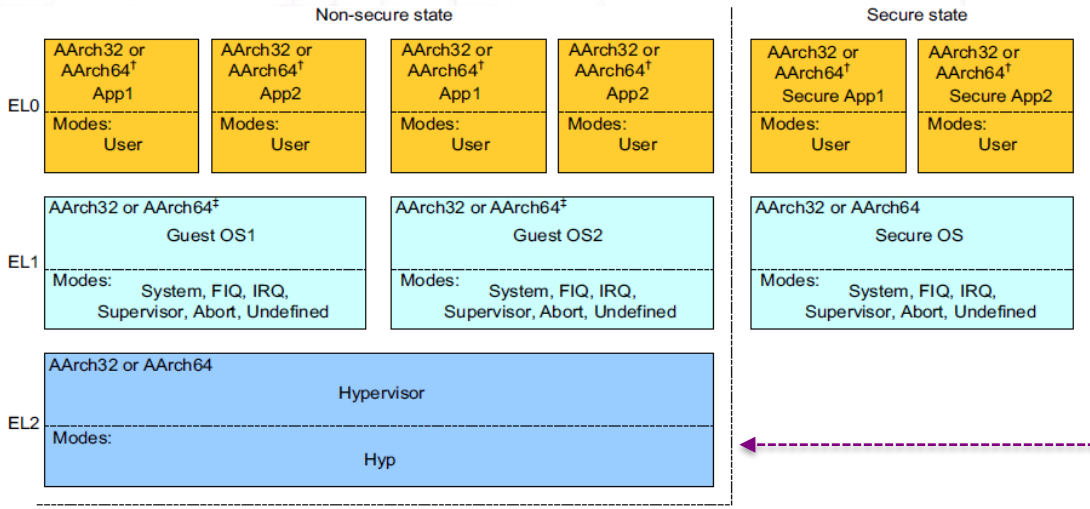
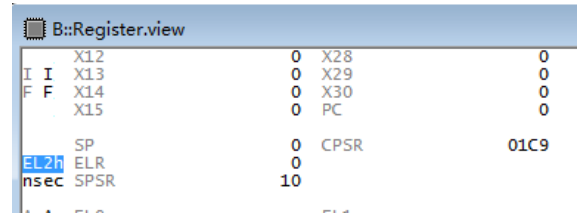


Figure 3-1 ARMv8 security model when EL3 is usi



启动阶段，运行在EL2/nosec（或hypervisor）

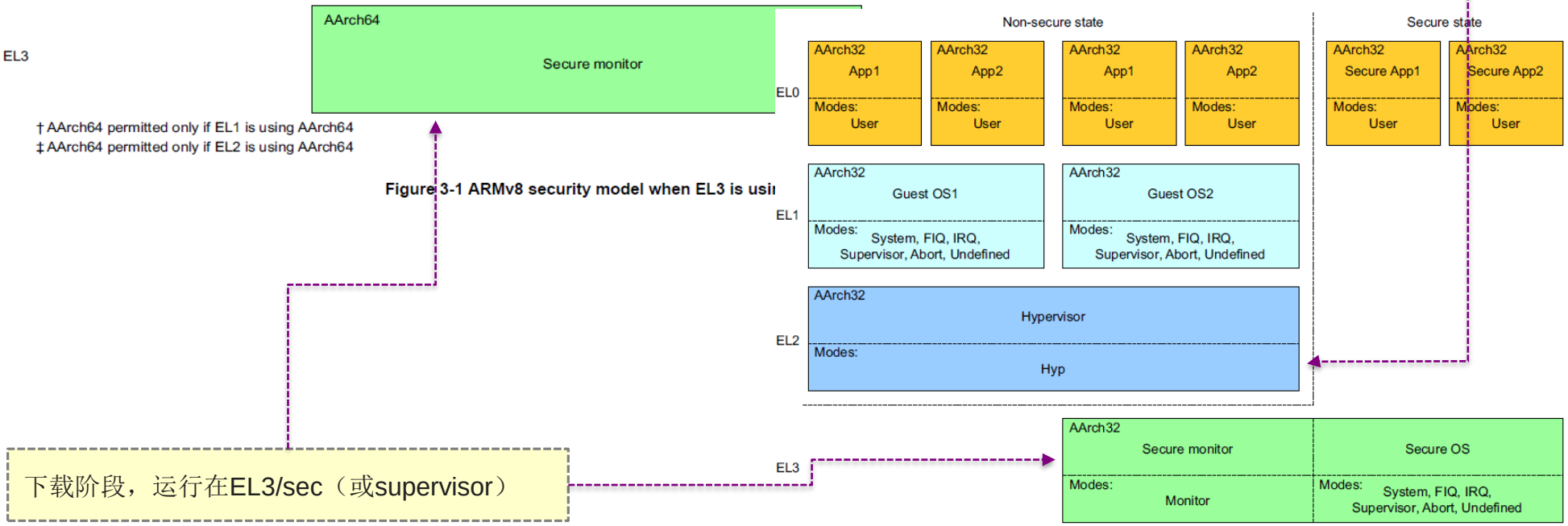


Figure 3-2 ARMv8 security model when EL3 is using AArch32

下载阶段，运行在EL3/sec（或supervisor）

03

内存布局

内存布局

Domains	Start address	End address(Not include)	Size	Size_readable	单位	
AP	80008000			8	M	Kernel Image
DTB	81000000			64	K	DTB是由DTS生成的binary，需要以固定地址开始。大小1M内
RamDisk	81100000			3.5	M	固定地址开始，最终会被释放
SML	81C00000	81D00000	100000	1	M	开始地址需4M对齐，
TOS	81D00000	82000000	300000	3	M	
smem	87800000	88000000	800000	8	M	sipc-mem(AP and CP share)　Bringup版本沿用之前地址
CP	89600000	8DD00000	5800000	88	M	CP MODEM　Bringup版本沿用之前地址
UBOOT	8F000000	8FFC0000	FC0000	15.75	M	For sysdump，Uboot未裁剪
KernelSW				1	M	mem_map size 128m/4K*32=1M and others etc.

Figure 5 Memory Layout 地址 规划示例

board/spreadtrum/<board>/Kconfig
config SYS_TEXT_BASE
default 0x8F000000

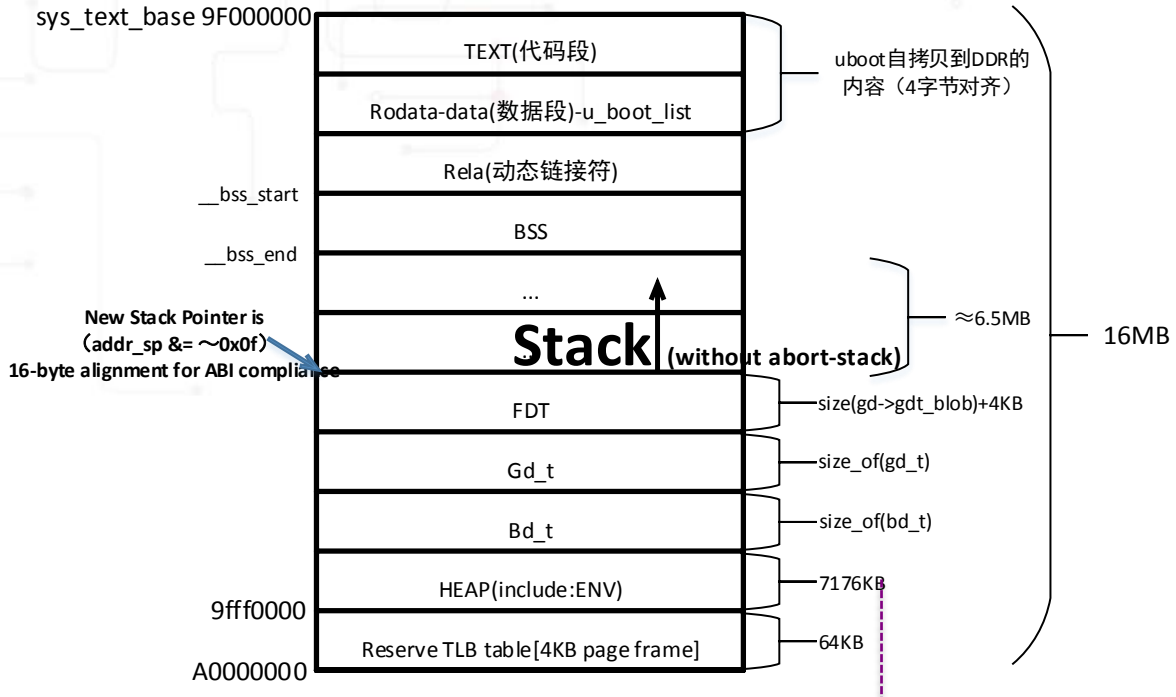
系统硬件资源规划一般在项目初期就已经确定，对于DDR的地址空间使用规划举例如上图所示。在bootloader阶段，uboot可以直接使用除安全限制外（SML/Tos）其它ddr地址空间。分配规则定义在内部dts里。如左图所示，模块可以在代码中，获取并直接使用，方式举例如下：

unsigned long buf_base, buf_size;

get_buffer_base_size_from_dt("heap@8", &buf_base, &buf_size); //heap 8对应logbuffer节点，地址空间范围[0x82000000, 0x8204FFFF]

```
reserved-memory {
    #address-cells = <1>;
    #size-cells = <1>;
    ranges;

    fastboot_reserved: fastbootbuffer@80000000{
        reg = <0x80000000 0x02000000>;
    };

    download_reserved: sparsebuffer@80000000{
        reg = <0x80000000 0x02000000>;
    };

    dl_alt1_reserved: alterbuffer1@80000000{
        reg = <0x80000000 0x02000000>;
    };

    dl_alt2_reserved: alterbuffer2@80000000{
        reg = <0x80000000 0x02000000>;
    };

    log_reserved: logbuffer@82000000{
        reg = <0x82000000 0x00050000>;
    };

    sml_reserved: sml-mem@82050000{
        reg = <0x82050000 0x00020000>;
    };

    tos_reserved: tos-mem@82070000{
        reg = <0x82070000 0x001e0000>;
    };

    secboot_arg_reserved: secboot-arg-mem@82250000{
        reg = <0x82250000 0xf00000>;
|== sl8563_2h10.dts 28:0 = 15:28 (aBIMtU) a:/project/unc_
    heap@8 {
        reg = <8>;
        label = "carveout_log";
        type = <2>;
        memory-region = <&log_reserved>;
    };
  };
};
|== sl8563_2h10.dts 110:0 = 15:50 (aBIMtU) a:/
```

内存布局（con’t）



Uboot自身运行空间，规划如上图所示。基本源用社区方案，未做改动。提供malloc/free动态内存申请接口。所使用的**堆**大小，由于全局的运行空间规划限制，通常会定义的比较小些，具体定义参见：

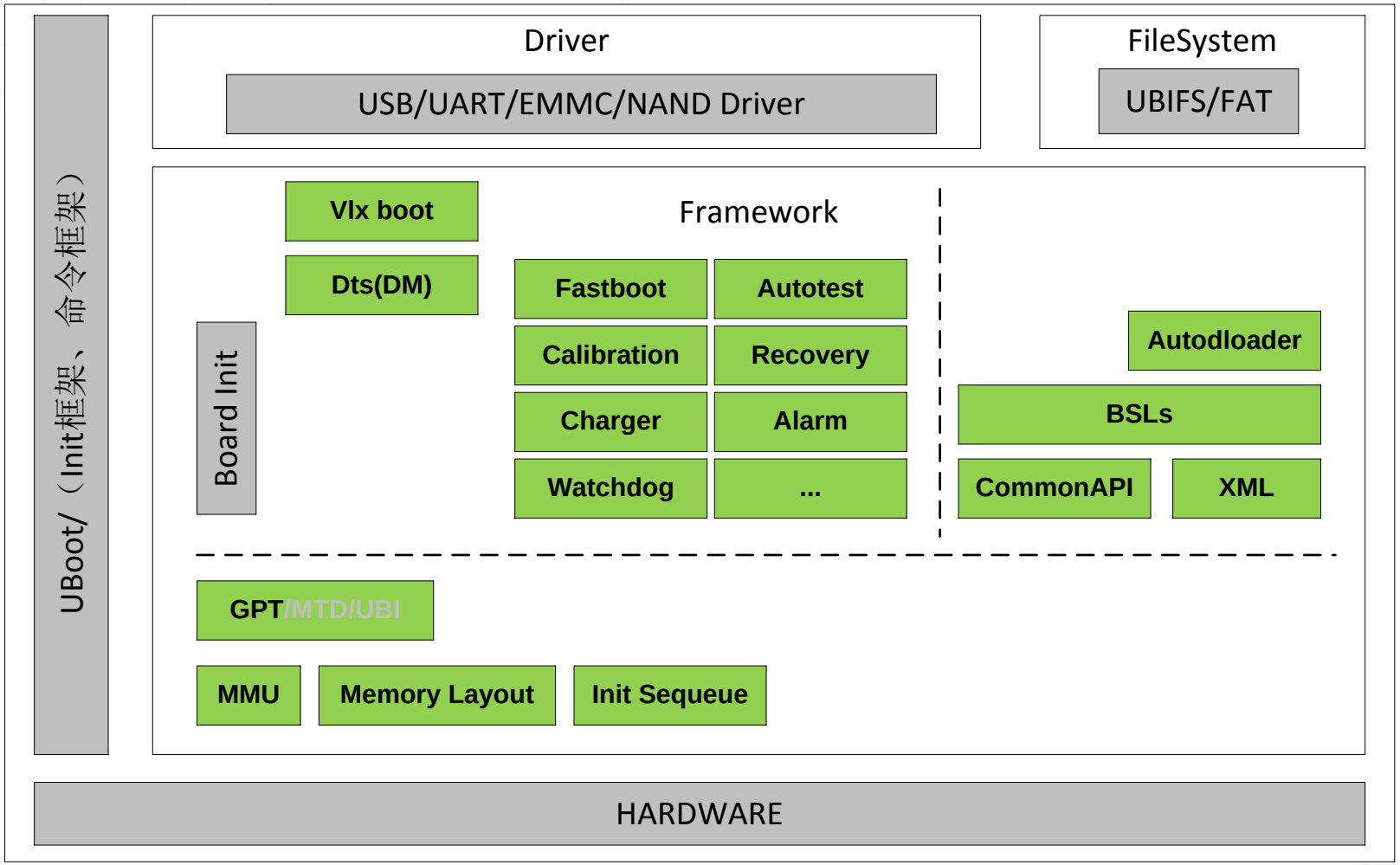
include/configs/<board>.h

```
/* Size of malloc() pool */
#define CONFIG_SYS_MALLOC_LEN                (CONFIG_ENV_SIZE + 3 * 1024 * 1024)
```

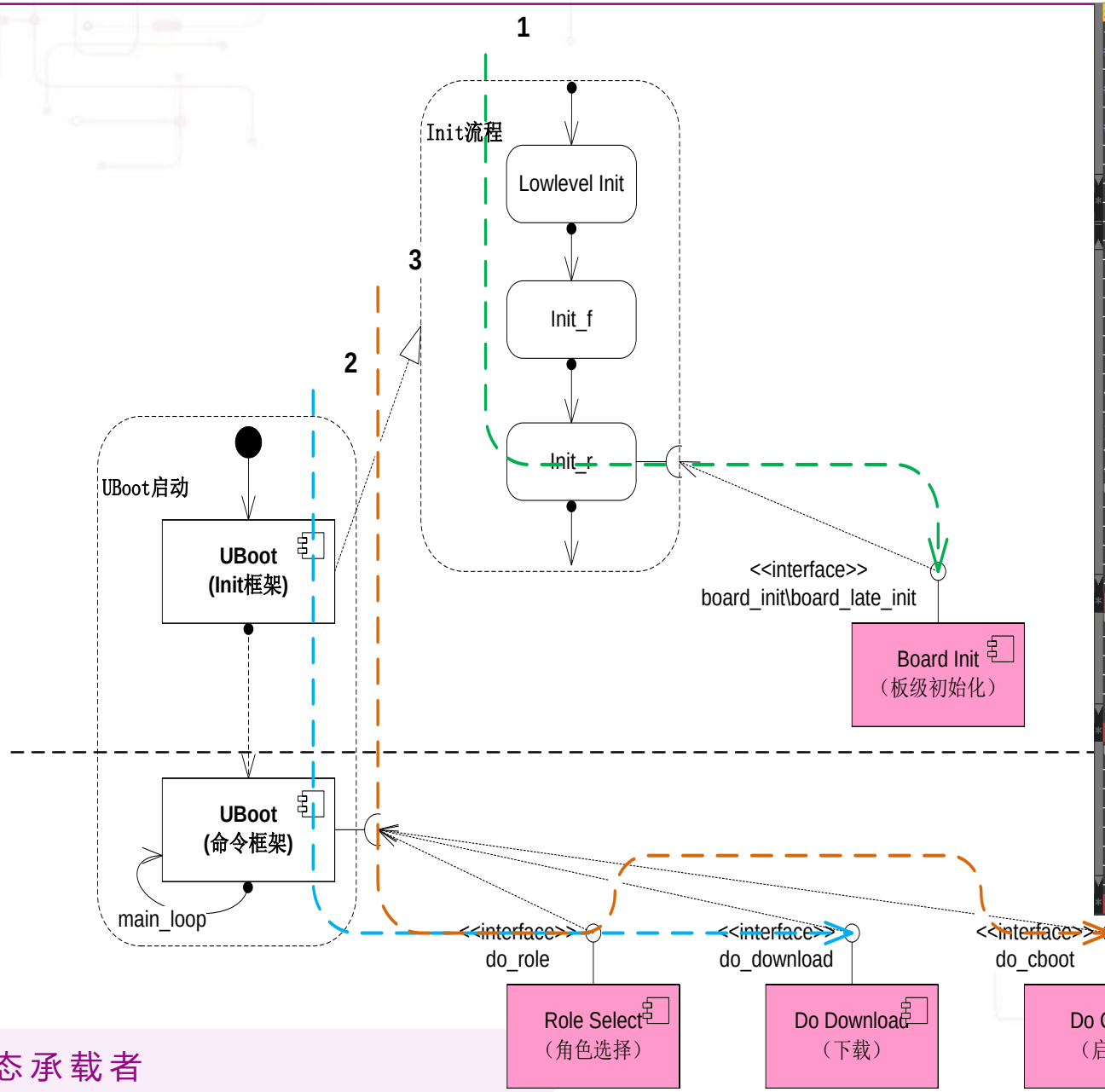
04

软件架构

软件架构


UBoot/（Init框架、命令框架）
Driver
USB/UART/EMMC/NAND Driver
FileSystem
UBIFS/FAT
Vlx boot
Dts(DM)
Framework
Fastboot
Autotest
Calibration
Recovery
Charger
Alarm
Watchdog
...
Board Init
Autodloader
BSLs
CommonAPI
XML
GPT/MTD/UBI
MMU
Memory Layout
Init Sequeue
HARDWARE

软件架构（con't）




```
int do_role(cmd_tbl_t *cmdtp, int flag, int argc, char *const argv[])
{
#ifdef CONFIG_ZEBU
    boot_mode_t boot_role = BOOTLOADER_MODE_LOAD;
#else
    boot_mode_t boot_role = get_boot_role();
#endif
    switch(boot_role) {
        case BOOTLOADER_MODE_DOWNLOAD:
            setenv("bootdelay", "0");
            setenv("bootcmd", "download");
            debugf("Get chipram env mode %x,go download\n", boot_role);
            break;
        case BOOTLOADER_MODE_LOAD:
            setenv("bootcmd", "cboot");
            debugf("Get chipram env mode %x,go cboot\n", boot_role);
            break;
        default :
            printf("unkown uboot role ,nothing to do in preboot\n");
    }

    return 0;
}


U_BOOT_CMD(
        role, CONFIG_SYS_MAXARGS, 1, do_role,
        "choose the role of spreadtrum uboot",
        "two roles:dloader and cboot\n"
    );
|=% cmd_role.c 36:0 = 13:31 (aBMtUV) +ense_dev/u-boot15/common/cmd_role.c
U_BOOT_CMD(
    download,  CONFIG_SYS_MAXARGS,1, do_download,
    "download mode",
    "choose to enter download mode\n"
    "it is used for downloading firmwares to storage in factory or resea
);
|=% cmd_download.c 189:0 = 13:32 (aBMtUV) +u-boot15/common/cmd_download.c
U_BOOT_CMD(
        cboot, CONFIG_SYS_MAXARGS, 1, do_cboot,
        "choose boot mode",
        "mode: \nrecovery, fastboot, dloader, charge, normal, vlx, cal
        "cboot could enter a mode specified by the mode descriptor.\n"
        "it also could enter a proper mode automatically depending on
        "the environment\n"
    );
|=% cmd_cboot.c 302:11 = 13:33 (aBMtUV) +_dev/u-boot15/common/cmd_cboot.c
```

软件架构（con't）--初始化（1）

◆底层板级初始化，分为如下3个阶段：

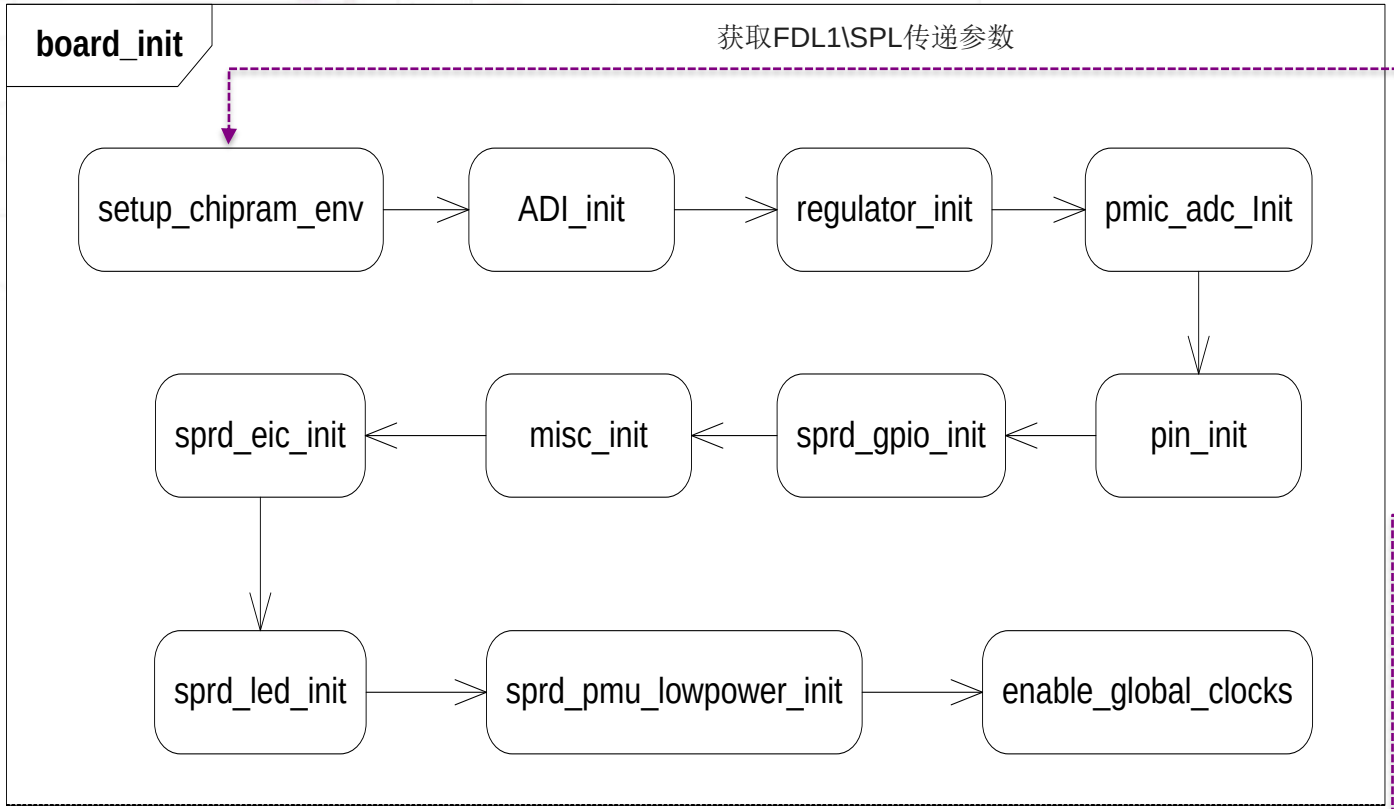
- Lowlevel Init， 主要完成cache使能及栈设置等SOC底层初始化，由汇编来完成。
- Init_f，c代码，完成部分外围（如timer、serial等）初始化，并重定位代码段，并**清空bss**，初始化全局变量gd，重新设置堆栈等，为后面运行创造完整的c运行环境。
- Init_r，c代码，完成外围设备初始化,调用展锐私有初始化接口(board_init\board_late_init)

: arch/arm/cpu/armv8/Start.s

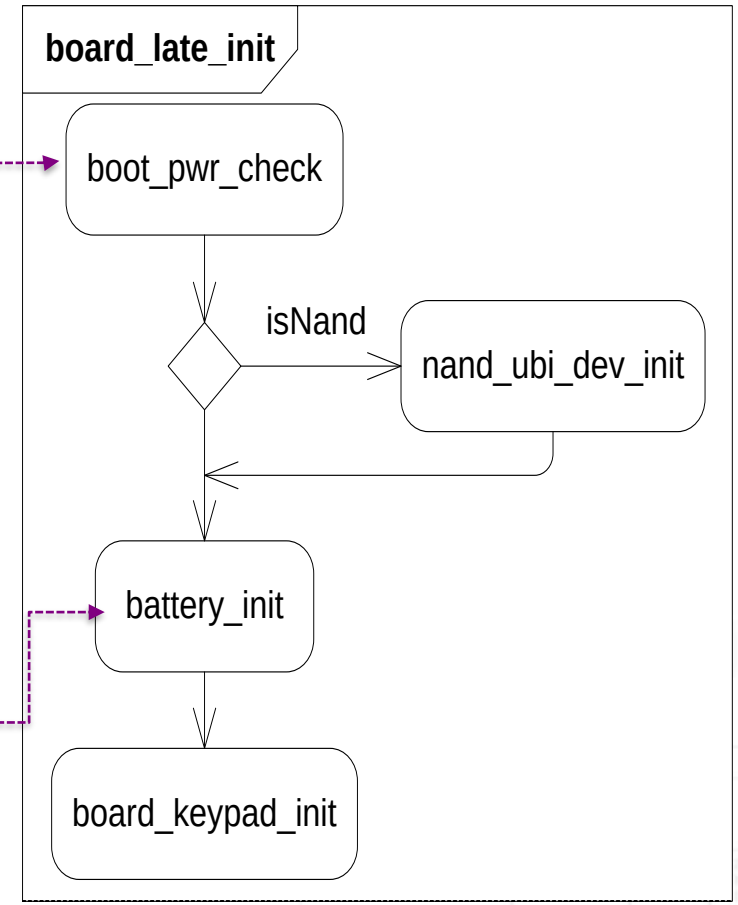
board_init_r(): arch/arm/lib/board.c

board_init_f(): arch/arm/lib/board.c

软件架构（con't）--初始化（2）

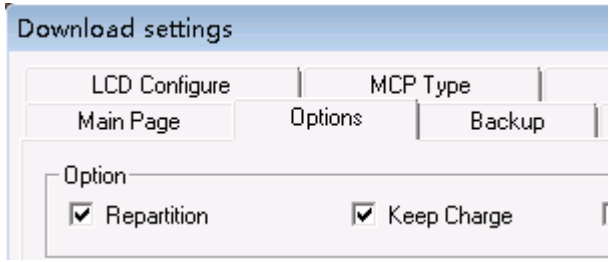



```
#define CHIPRAM_ENV_LOCATION 0x82000000
typedef struct chipram_env {
  u32 magic; //0x43454e56
  boot_mode_t mode;
  u64 dram_size;
  u32 keep_charge;
...
}chipram_env_t;
```

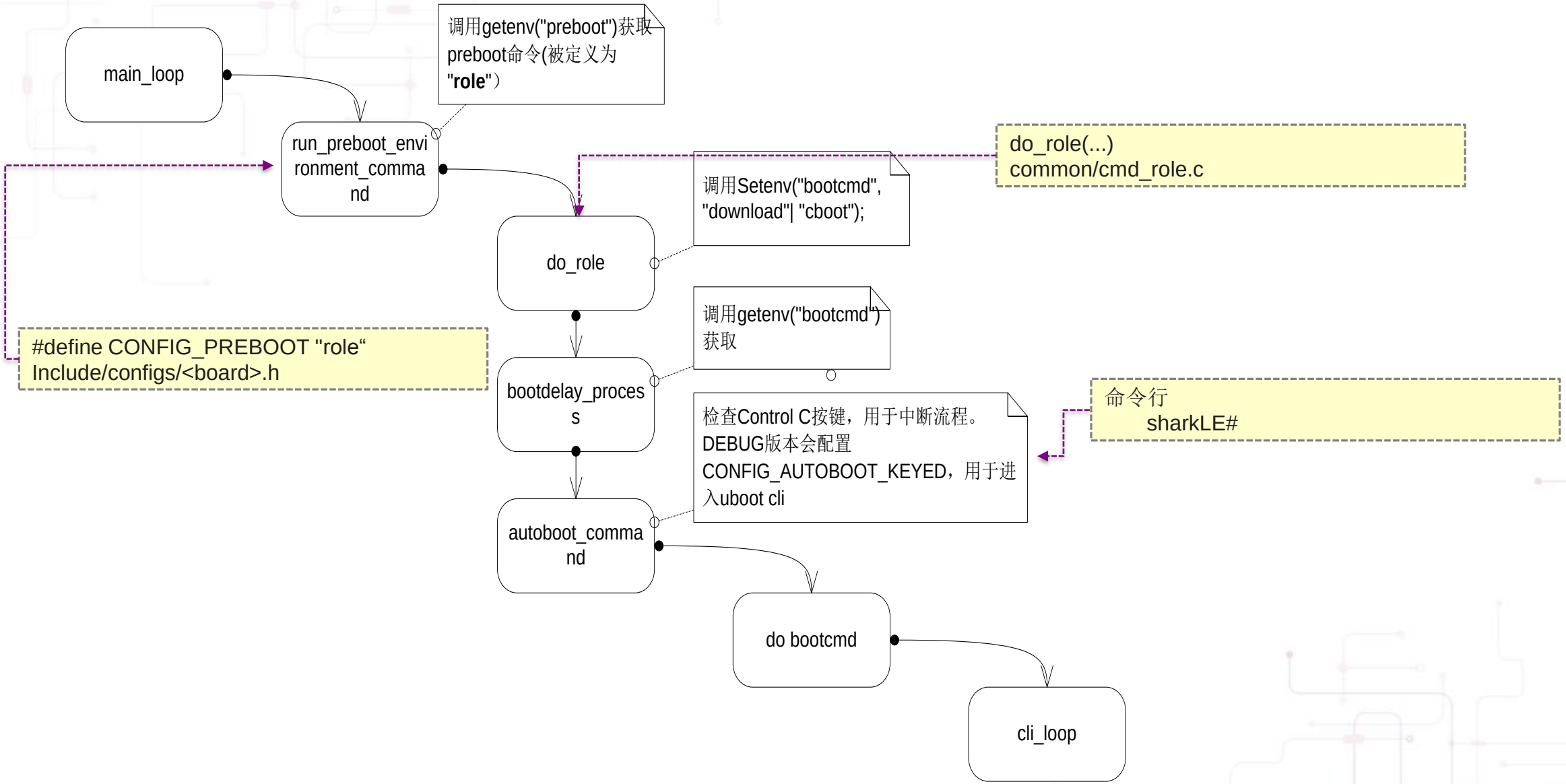


开机键（PBInt）检测

下载 低电检测



软件架构（con't）--命令框架



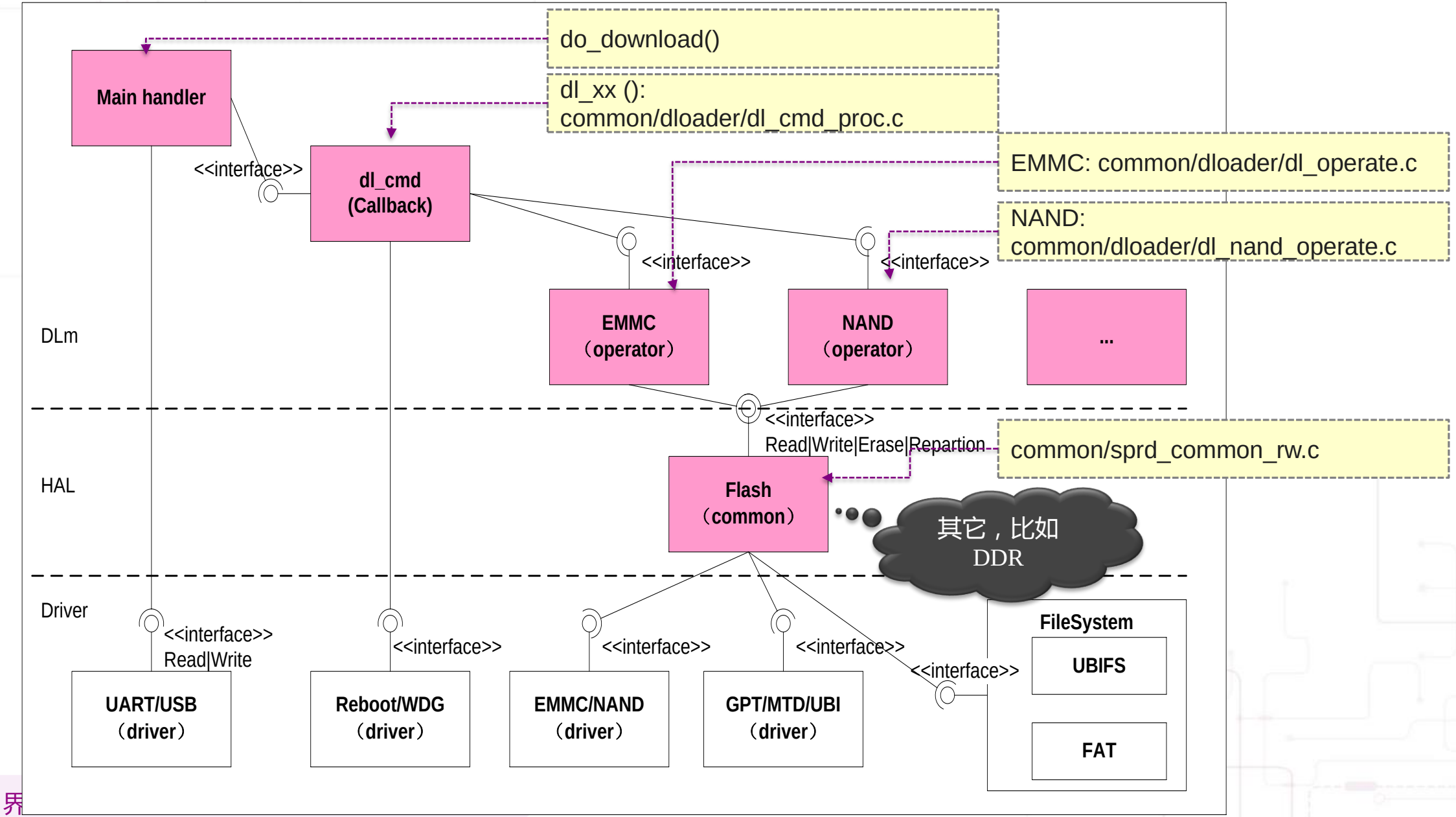
软件架构（con’t）--下载流程

◆下载流程：

- 初始化，从init_r进入命令处理框架。
- 命令框架，调用Role Select模块注册的回调接口，处理由FDL1传递环境参数，选择进入下载模式。
- 下载模块，根据下载协议与PCTool交互，完成将image下载到对应的flash分区。

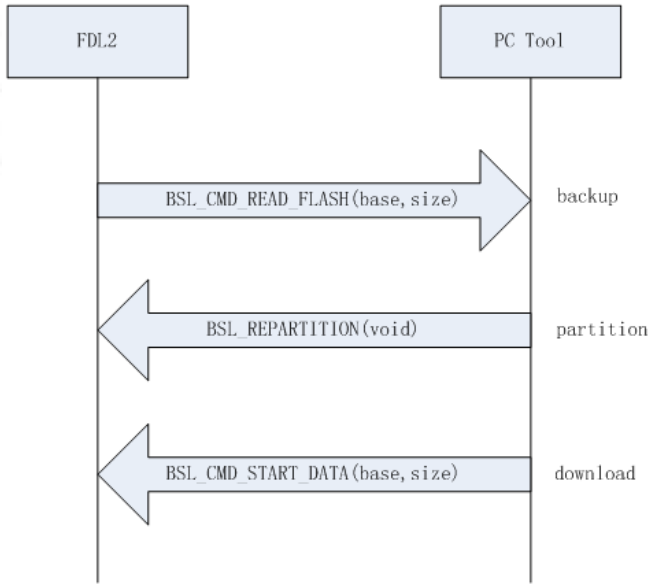
do_download():common/cmd_download.c

软件架构（con't）--下载流程（con't）



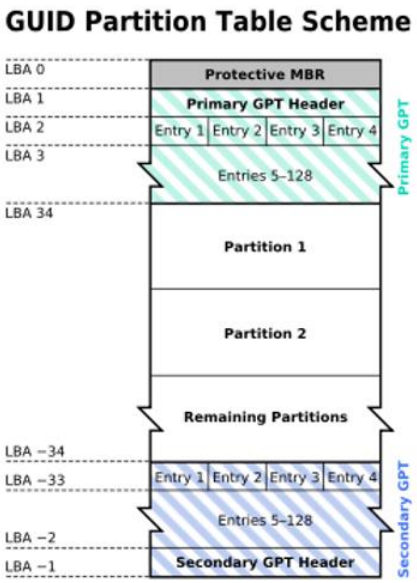
软件架构（con't）--下载流程（con't）

工具下载image前会先进行backup，然后再进行重分区，eMMC采用GPT分区表格式，如左图所示：



- eMMC采用GPT分区表格式，如左图所示：
- Nand采用Mtd，定义在：include/configs/<board>.h

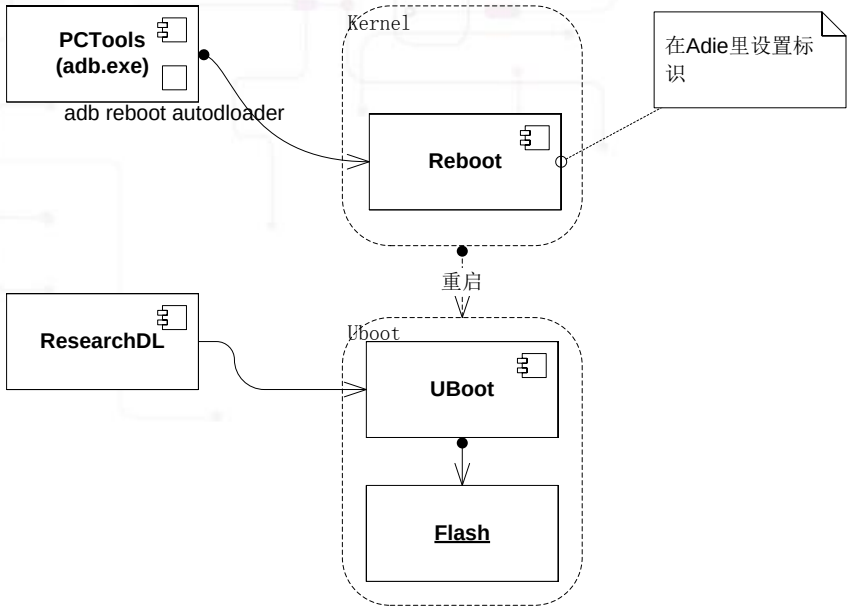
#define MTDPARTS_DEFAULT "mtdparts=sprd-nand:256k(splloader),1280k(uboot),768k(sml),1024k(trustos),-(**ubipac**)"




```
<Partitions>
    <!-- size unit is MBytes -->
    <Partition id="prodnv" size="10"/>
    <Partition id="miscdata" size="1"/>
    <Partition id="misc" size="1"/>
    <Partition id="trustos_a" size="6"/>
    <Partition id="trustos_b" size="6"/>
    <Partition id="sml_a" size="1"/>
    <Partition id="sml_b" size="1"/>
    <Partition id="uboot_a" size="1"/>
    <Partition id="uboot_b" size="1"/>
    <Partition id="uboot_log" size="4"/>
    <Partition id="logo" size="8"/>
    <Partition id="fbootlogo" size="8"/>
    <Partition id="l_fixnv1" size="2"/>
    <Partition id="l_fixnv2" size="2"/>
    <Partition id="l_runtimenv1" size="2"/>
    <Partition id="l_runtimenv2" size="2"/>
    <Partition id="vbmeta_vendor_a" size="1"/>
    <Partition id="vbmeta_vendor_b" size="1"/>
    <Partition id="vbmeta_product_a" size="1"/>
    <Partition id="vbmeta_product_b" size="1"/>
    <Partition id="userdata" size="0xFFFFFFFF"/>
</Partitions>
```

ResearchDownload_R24.20.1001\Bin\ImageFiles_Down...\<...>.xml

软件架构（con’t）--下载流程（autodloader）



工具执行adb reboot autodloader，Kernel reboot模块处理在Adie上设置进autodloader的标识位。

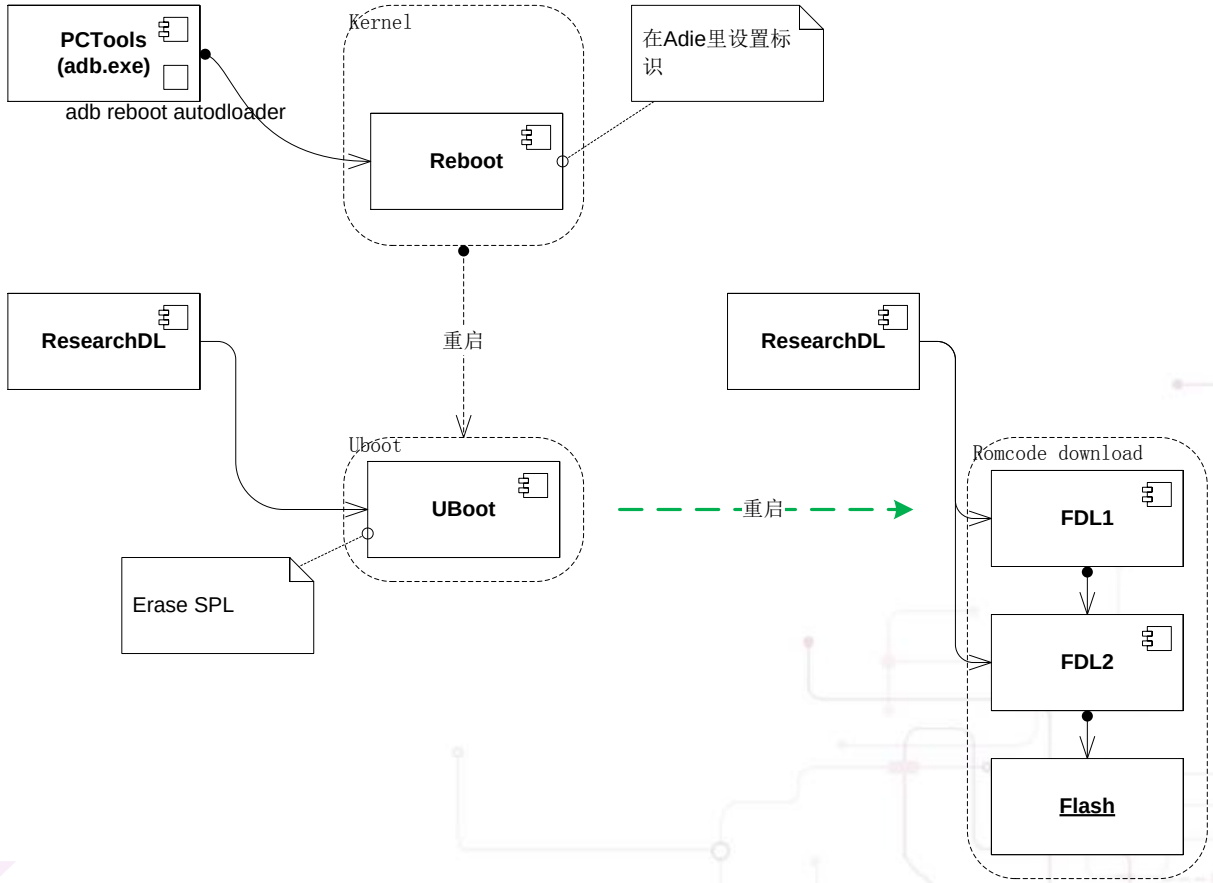
UBoot流程：

方案1：

模拟romcode发送ver并与Tool建立连接，接收FDL1/FDL2，bypass。接收后续的image直接写入flash。

方案2：

擦除SPL，并重启。后续romcode开机发现SPL无效，走下载流程。与Tool交互下载image到flash。



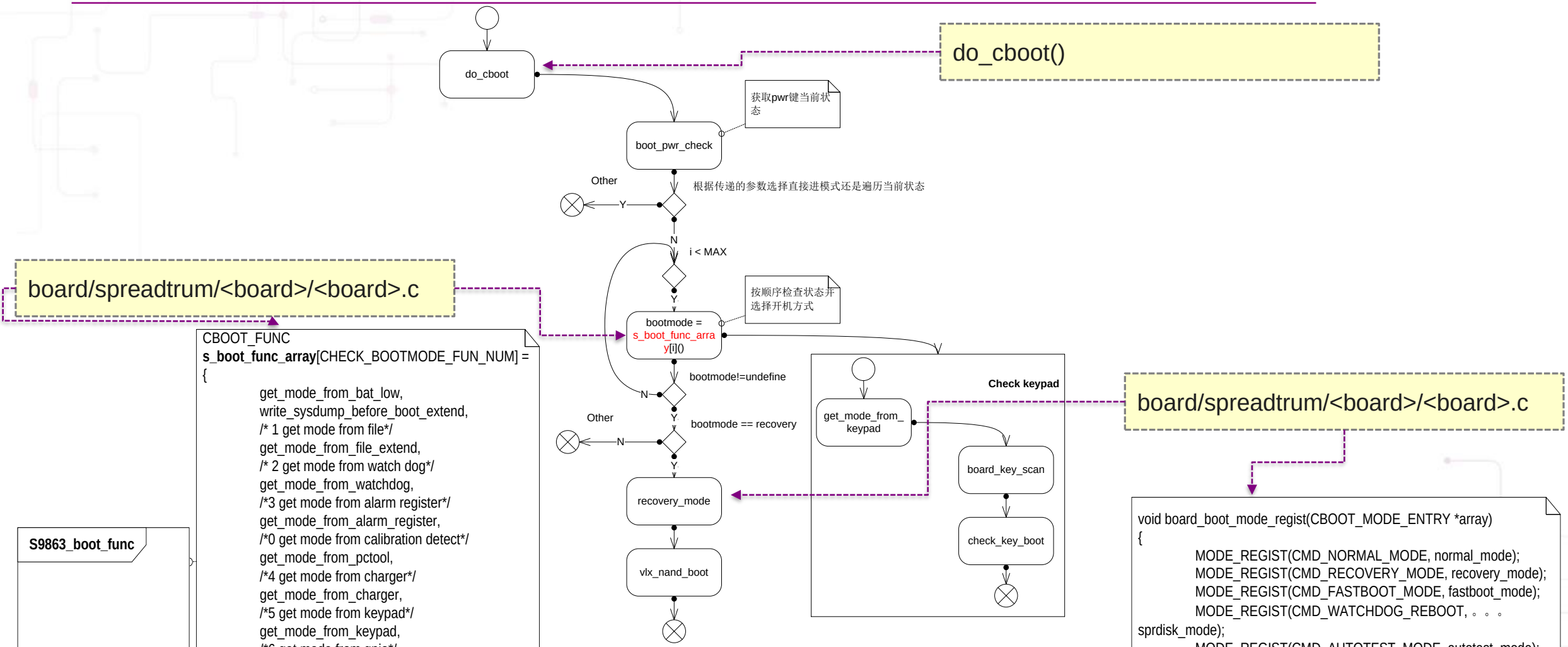
软件架构（con't）--启动流程

◆启动流程：

- 初始化，从init_r进入命令处理框架。
- 命令框架，调用Role Select模块注册的回调接口，由SPL.bin传递的参数决定进入启动模式Do Cboot。
- 根据配置或外设状态（比如power key）等选择进入指定模式。根据模式定义与PCTool交互完成校准或者从Flash加载kernel、ramdisk及dtb、dtbo等，跳转进入kernel。根据配置负责启动SP/PUBCP等Subsystem。

do_cboot():common/cmd_cboot.c

软件架构（con't）--启动流程（con't）



do_cboot()

board/spreadtrum/<board>/<board>.c

board/spreadtrum/<board>/<board>.c

S9863_boot_func

```
CBOOT_FUNC
s_boot_func_array[CHECK_BOOTMODE_FUN_NUM] =
{
        get_mode_from_bat_low,
        write_sysdump_before_boot_extend,
        /* 1 get mode from file*/
        get_mode_from_file_extend,
        /* 2 get mode from watch dog*/
        get_mode_from_watchdog,
        /*3 get mode from alarm register*/
        get_mode_from_alarm_register,
        /*0 get mode from calibration detect*/
        get_mode_from_pctool,
        /*4 get mode from charger*/
        get_mode_from_charger,
        /*5 get mode from keypad*/
        get_mode_from_keypad,
        /*6 get mode from gpio*/
        get_mode_from_gpio_extend,

        /*shutdown device*/
        //get_mode_from_shutdown,
        0
};
```

S9863_boot_mode

```
void board_boot_mode_regist(CBOOT_MODE_ENTRY *array)
{
        MODE_REGIST(CMD_NORMAL_MODE, normal_mode);
        MODE_REGIST(CMD_RECOVERY_MODE, recovery_mode);
        MODE_REGIST(CMD_FASTBOOT_MODE, fastboot_mode);
        MODE_REGIST(CMD_WATCHDOG_REBOOT, 。。。
sprdisk_mode);
        MODE_REGIST(CMD_AUTOTEST_MODE, autotest_mode);
        return ;
}
```

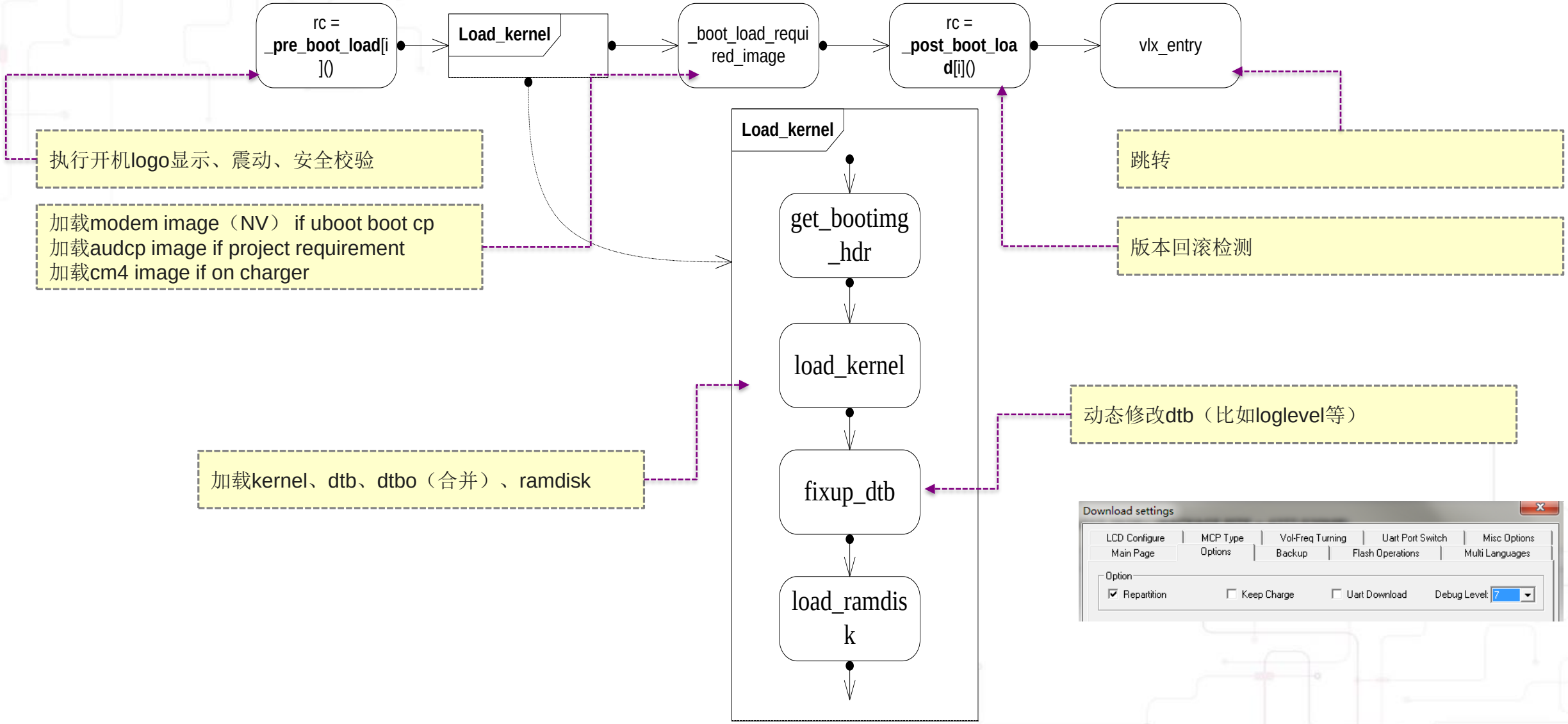
软件架构（con't）--启动流程 – 模式选择

- get_mode_from_bat_low 低电量检测
- write_sysdump_before_boot_extend 判断并处理sysdump
- get_mode_from_miscdata_boot_flag 判断并处理Firstmode
- get_mode_from_file_extend 从misc分区读取预设模式
- get_mode_from_watchdog 从Adie寄存器获取预设模式
- get_mode_from_alarm_register 从Adie寄存器获取预设模式
- get_mode_from_pctool 通过USB与Ptool交互确定是否进入cali或native mmi（bbat/autotest）
- get_mode_from_charger 进充电
- get_mode_from_keypad 判断并处理按键
- get_mode_from_gpio_extend 判断并处理gpio

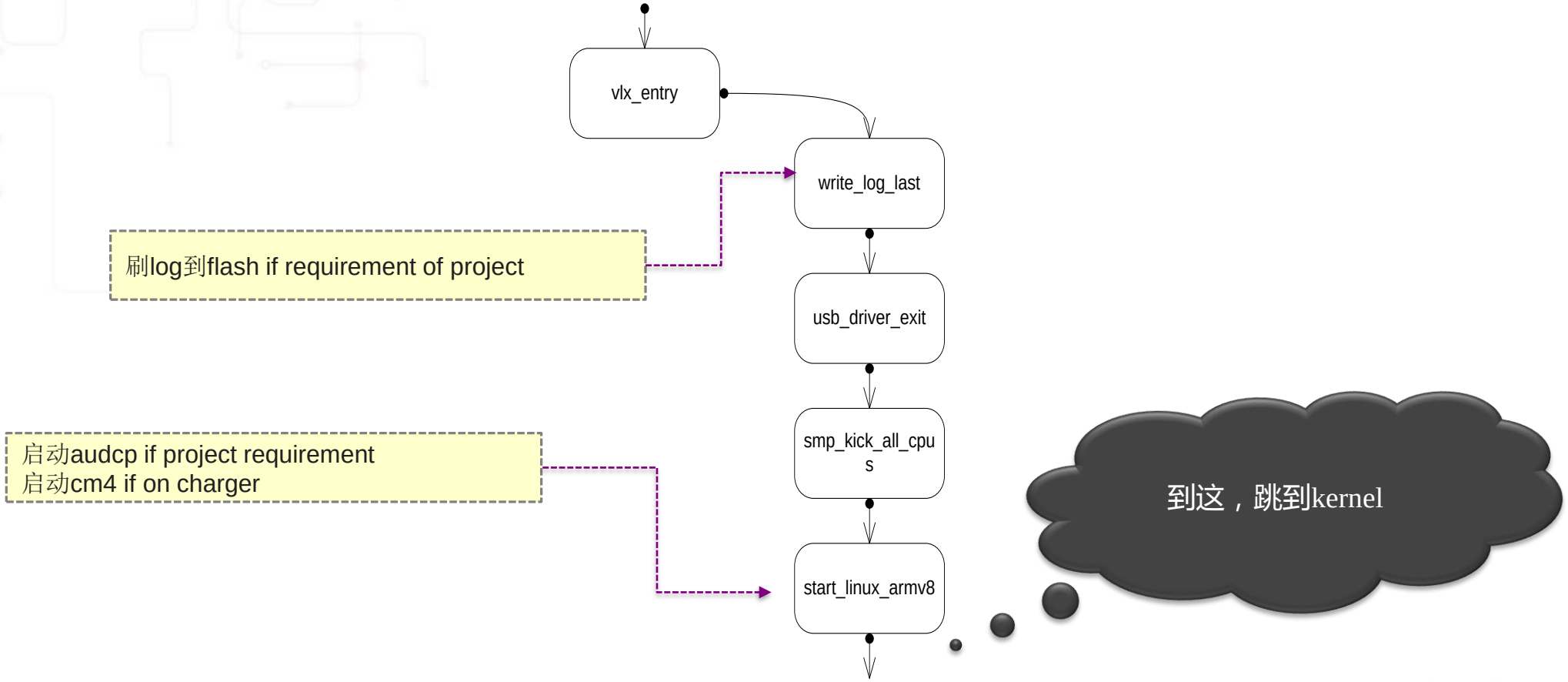
软件架构（con't）--启动流程 – 模式处理

模式	介绍
Fastboot	fastboot是一种比recovery更底层的刷机模式。进入Fastboot模式命令行：adb reboot fastboot
Autodloader	uboot进入autoloader模式后，会擦除SPL，重启，利用romcode对spl完整性校验，校验失败走下载流程。 进入Autodloader模式命令行: adb reboot autodloader
Systemdump	系统异常重启，uboot根据复位状态，进入systemdump处理流程。常见引发systemdump的原因：wdg复位，kernel panic等等
normal	正常开机模式
recovery	恢复模式，是一种可以对安卓内部的数据修改或系统进行升级的模式。
charge	充电模式
engtest	工程模式（EngineerMode，简写为EngMode），工程师用来调试底层硬件的各项参数的工具，通过暗码的方式进入，完成对电话相关参数的设置、网络相关的设置、调试手段的设置、系统信息的读取等；它不依赖于上层，可以在上层应用尚未开发完毕或者有逻辑问题时，直接判断调试底层问题。
factorytest	工厂模式主要用于产线的快速测试一些基本的硬件，芯片，电话功能，OTG测试功能等。为了能够快速的开机，并没有启动android，只是在Linux kernel启动启动，通过加载一些必须的native service，启动一个叫factoryTest的进程。
calibration	校准模式，用于modem校准等
iq	通信测试，用于W-IQ通信校准
alarm	闹铃模式
sprdisk	Sprdisk模式，Sprdis:是基于buildroot和ltp开源项目，以及加入了自己开发的testcases，且自带交叉编译工具链的一个独立的工程（支持32位和64位）；目的是提供一个含有ltp测试套件的虚拟内存盘ramdisk.img作为根文件系统，在标准的linux环境下对手机的软硬件模块进行测试
autotest	BBAT模式
watchdog	检查PMIC WDG复位状态寄存器，再决定是否引导系统进入WDG模式
ap_watchdog	检查AP WDG寄存器，再决定是否引导系统进入WDG模式
panic_reboot	检查PMIC WDG复位状态寄存器，再决定是否引导系统进入panic模式

软件架构（con't）--启动流程 – 加载



软件架构（con't）--启动流程 – 跳转



05

相关文档

相关文档

- U-Boot客制化指导手册V1.2.docx

谢 谢

紫光展锐
UNISOC