Boot & Download Process

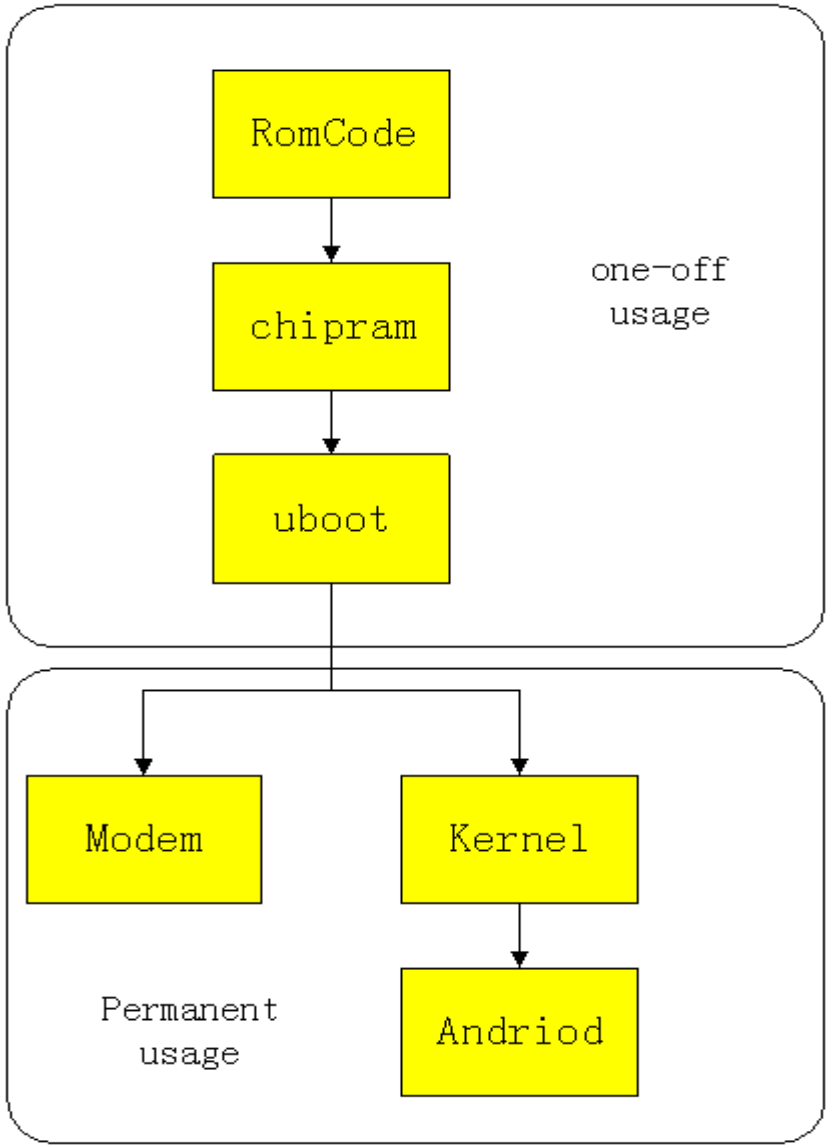
SPREADTRUM®

Agenda

- Overall Process
- Romecode
- 1st stage download FDL1
- 1st stage boot SPL
- U-boot Initialization Process
- U-boot Download Process
- U-boot Load Process

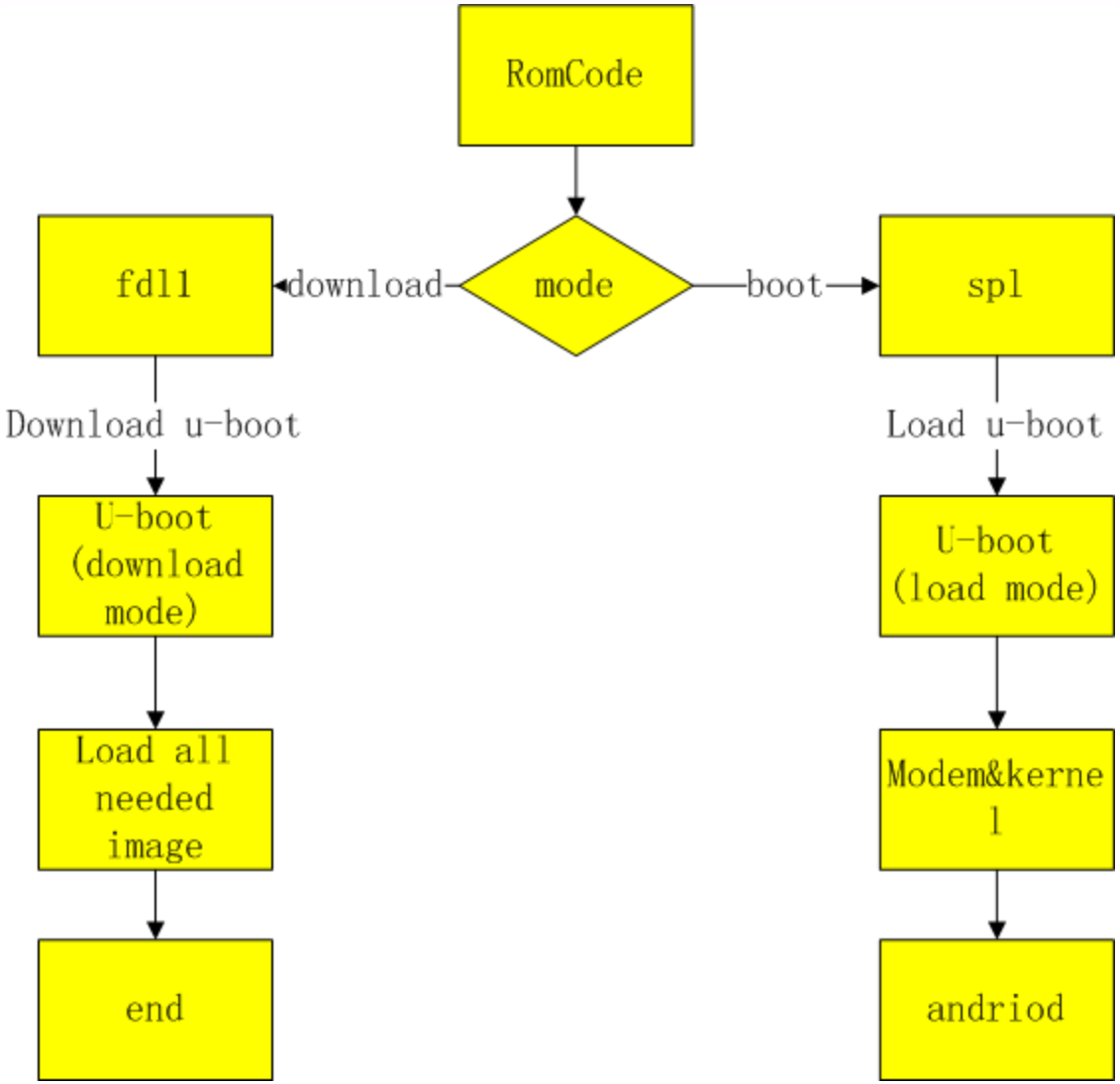
Overview

- Software Architecture



Overview

- Overall process of the whole system



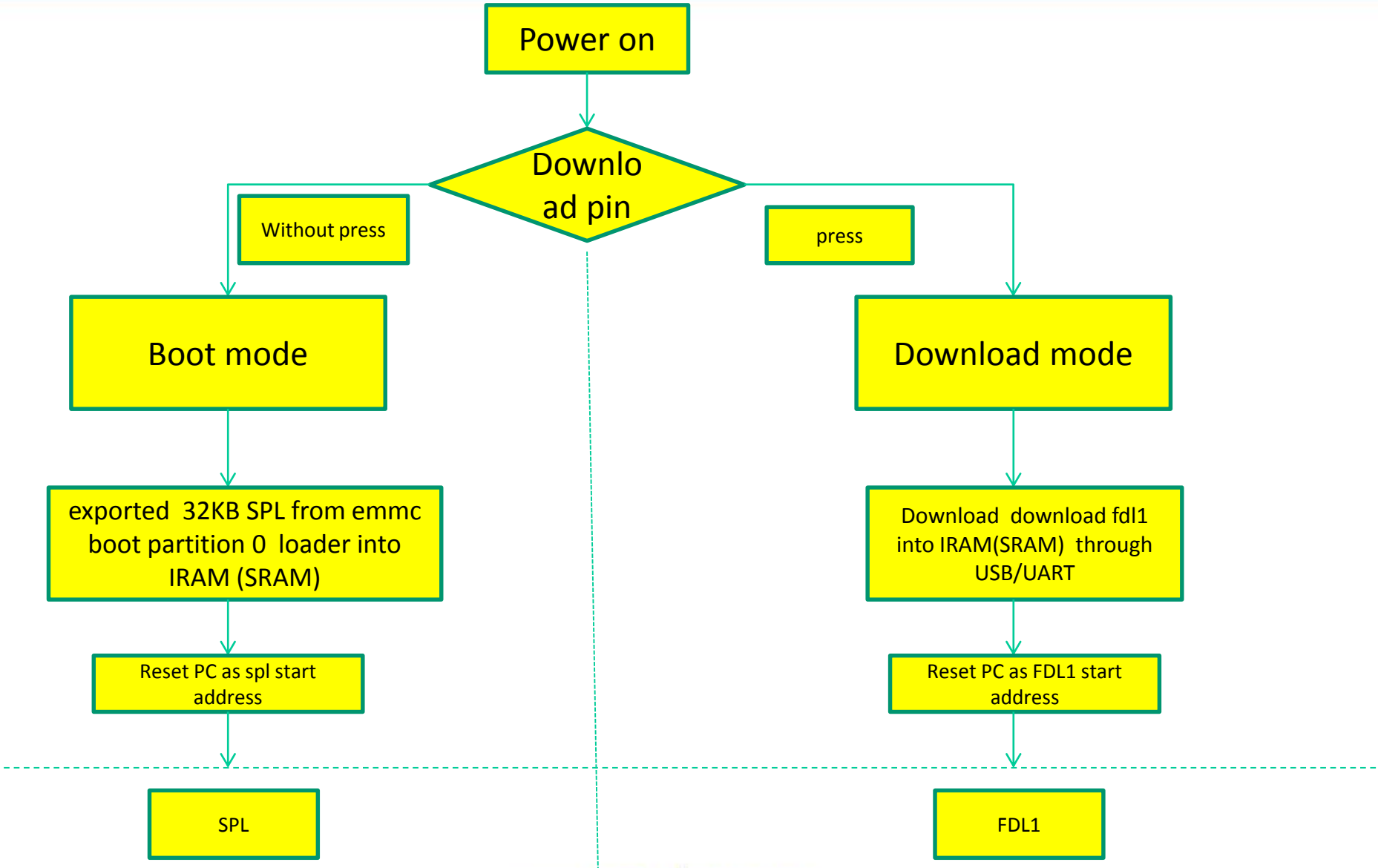
Agenda

- Overall Process
- Romecode
- 1st stage download FDL1
- 1st stage boot SPL
- U-boot Initialization Process
- U-boot Download Process
- U-boot Load Process

RomCode

- A small program stored and run in Rom. That means RomCode can never be modified and has the highest security level.
- As automatic run after power on, used for download fdl1 or boot spl.

RomCode



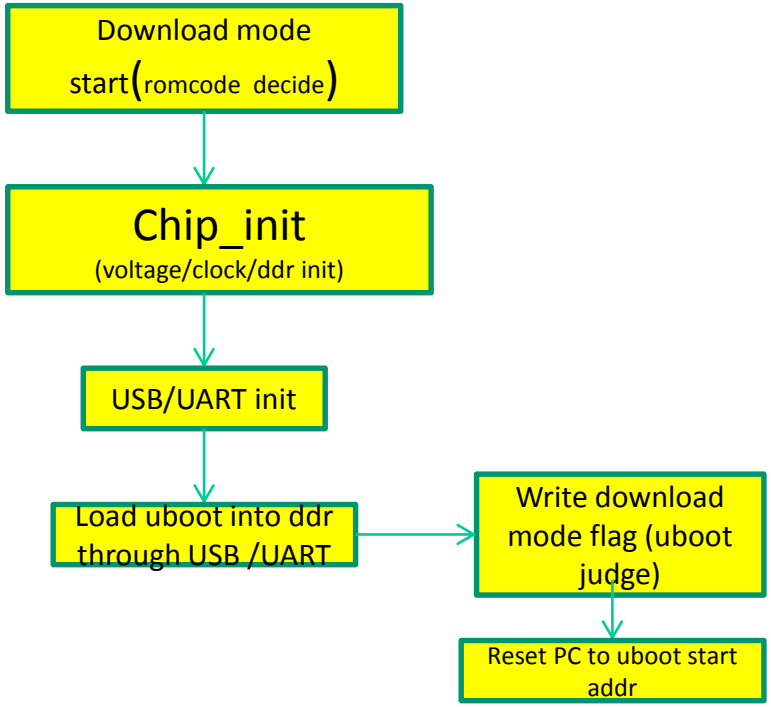
Agenda

- Overall Process
- Romecode
- 1st stage download FDL1
- 1st stage boot SPL
- U-boot Initialization Process
- U-boot Download Process
- U-boot Load Process

FDL1

- FDL1 and SPL collectively called Chipram.
- Main work of FDL1 is init voltage, clock, DDR then download FDL2 into DDR through USB/UART.
- FDL1 and SPL share the same code repository, but they are separate binary images.
- FDL1/SPL run entirely in Internal RAM(IRAM), typically they can use 32KB memory space.

FDL1 work flow.

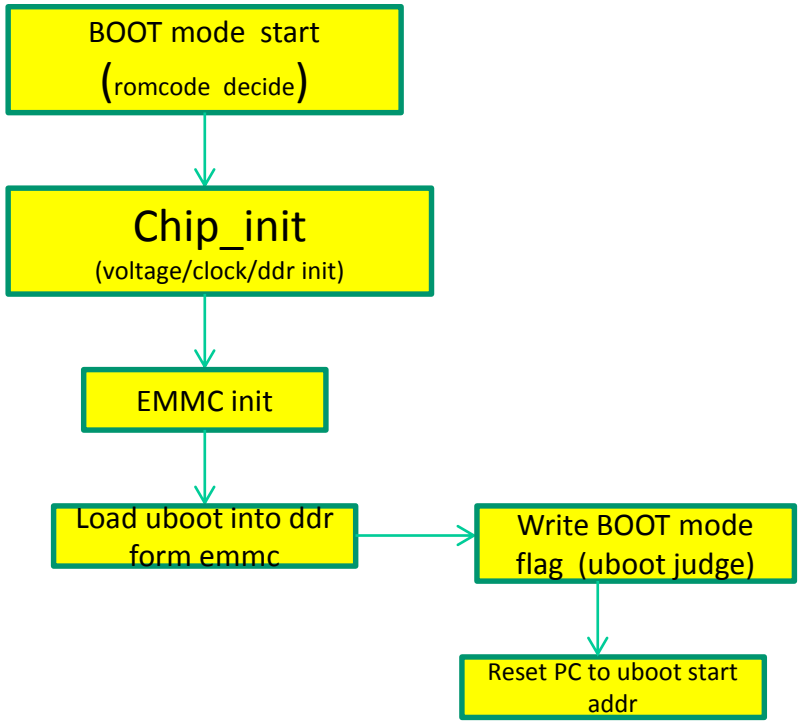


Agenda

- Overall Process
- Romecode
- 1st stage download FDL1
- 1st stage boot SPL
- U-boot Initialization Process
- U-boot Download Process
- U-boot Load Process

SPL

- Main work of SPL is init voltage, clock, ddr, eMMC then load u-boot into DDR from eMMC boot partition.



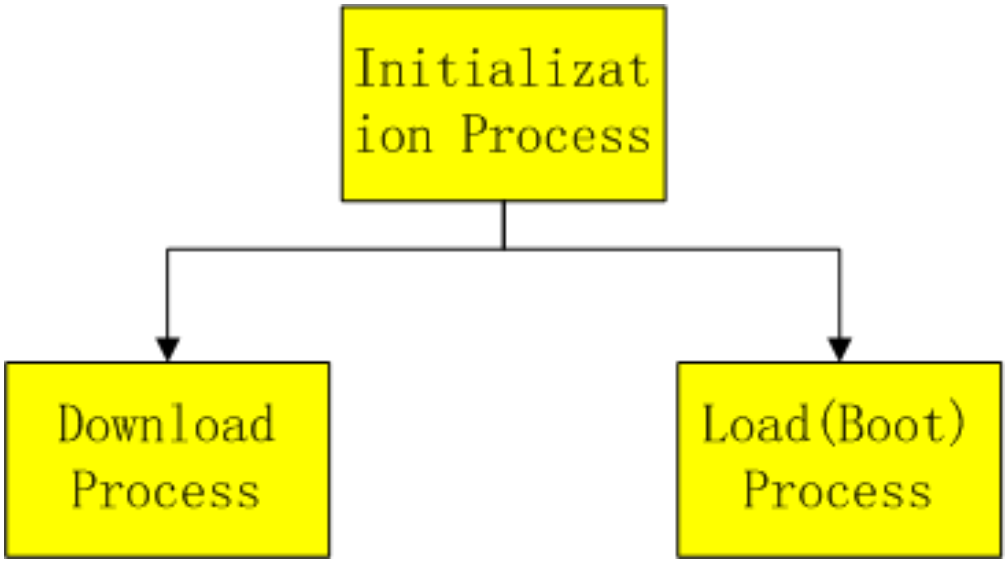
Agenda

- Overall Process
- Romecode
- 1st stage download FDL1
- 1st stage boot SPL
- U-boot Initialization Process
- U-boot Download Process
- U-boot Load Process

Initialization Process

Known that fdl2 and u-boot are the same

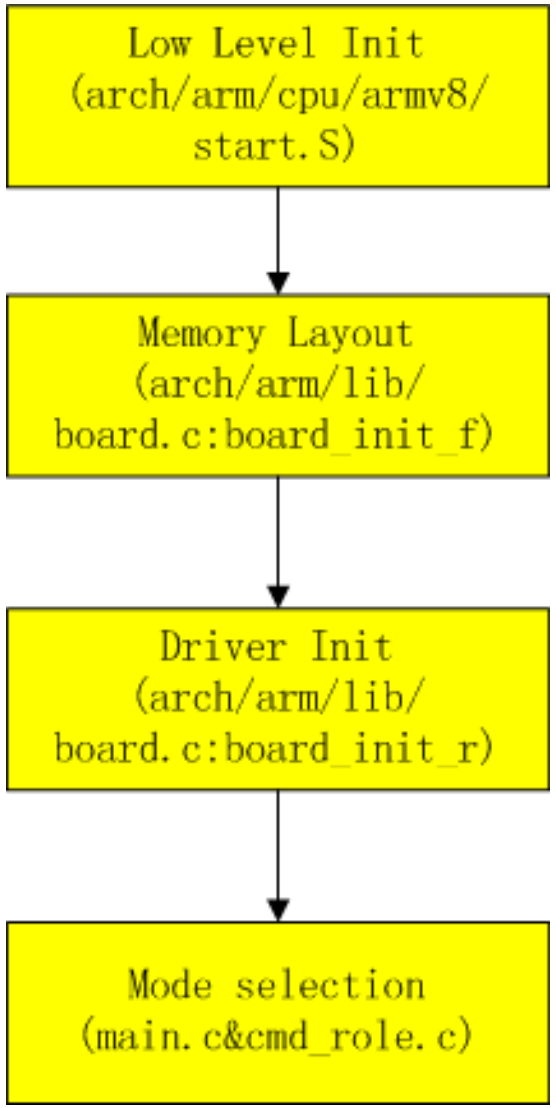
Divide u-boot into three parts and survey them respectively.



Initialization Process

Four tasks need to be done in Initialization process:

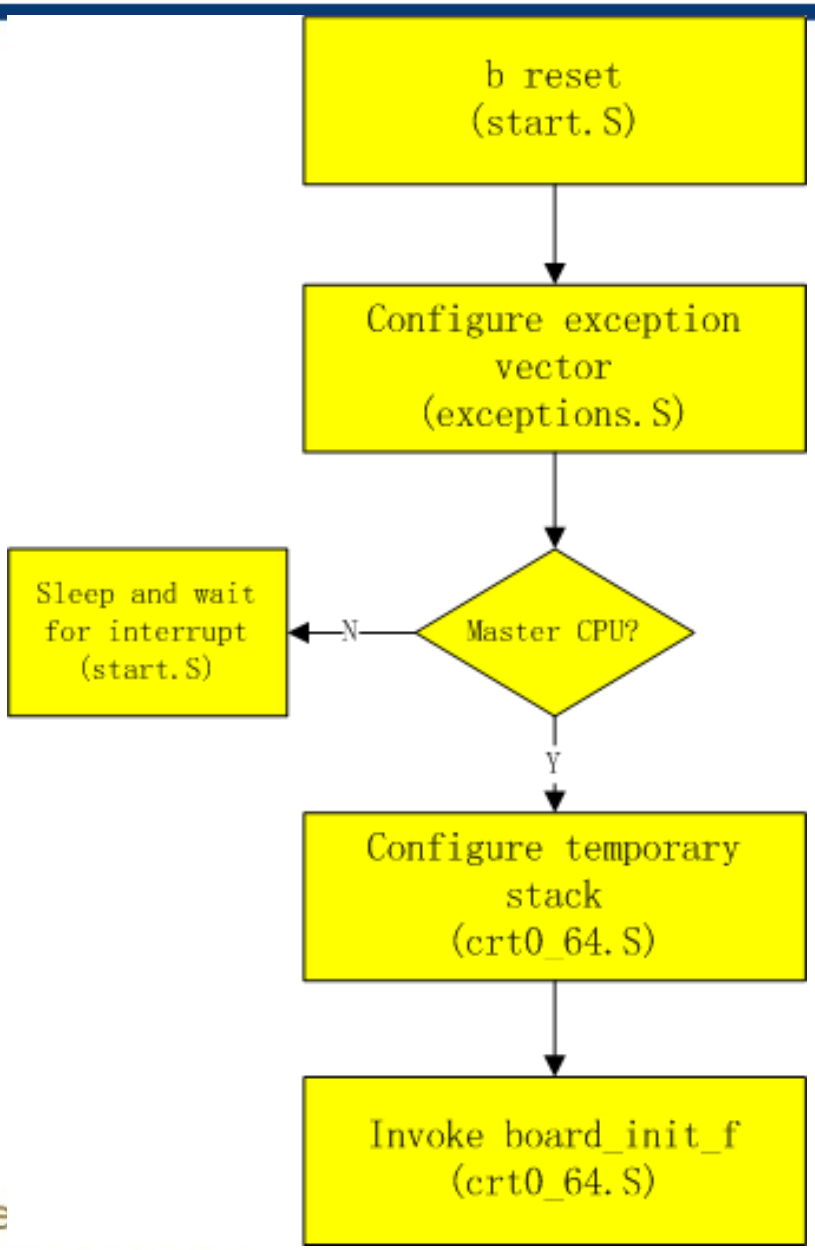
- Low level init: configuration of arm core register and preparation of C environment.
- Memory layout: determine position of each part of u-boot in RAM.
- Driver init: initialize all the drivers will be used later.
- Mode selection: select download mode or load mode



Initialization Process

Low level init

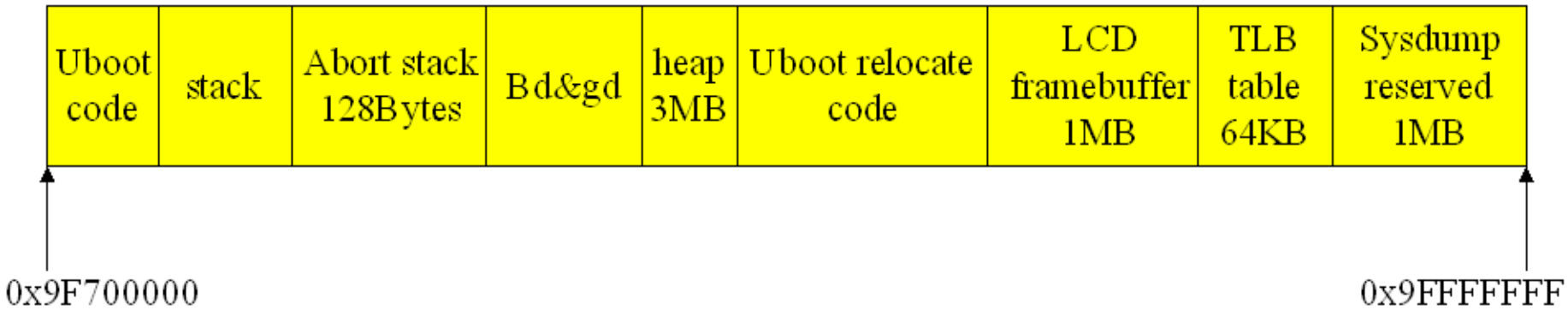
- "b reset" is the first line of code in u-boot.
- Master CPU(core0) move forward after low level init, meantime other cores sleep and wait kernel wake them up.
- Permanent stack will be established later, temporary stack only used in board_init_f function.



Initialization Process

Memory layout

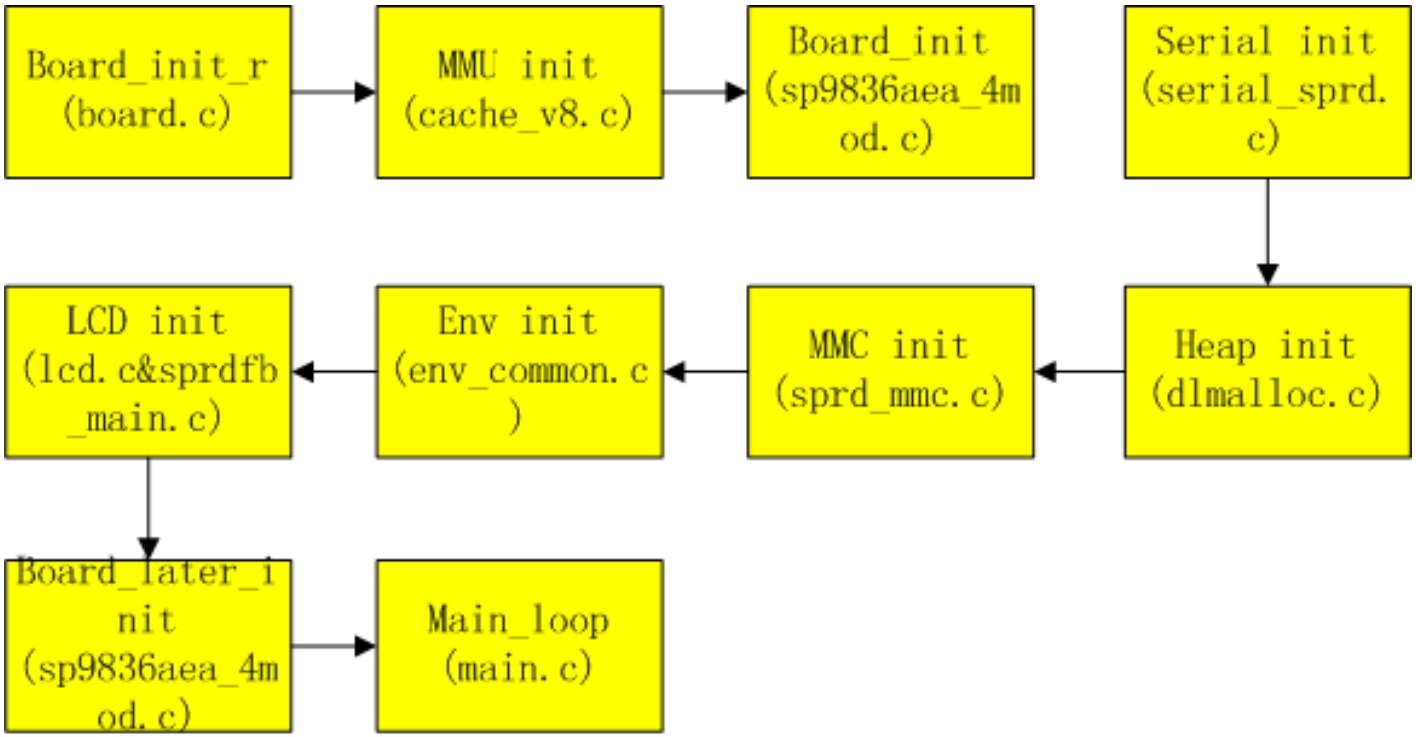
- All u-boot used memory are located between 0x9F700000 and 0xA0000000.
- For different boards memory layout may not be exactly the same.
- The last 1MB is reserved for system dump.



Initialization Process

Driver init

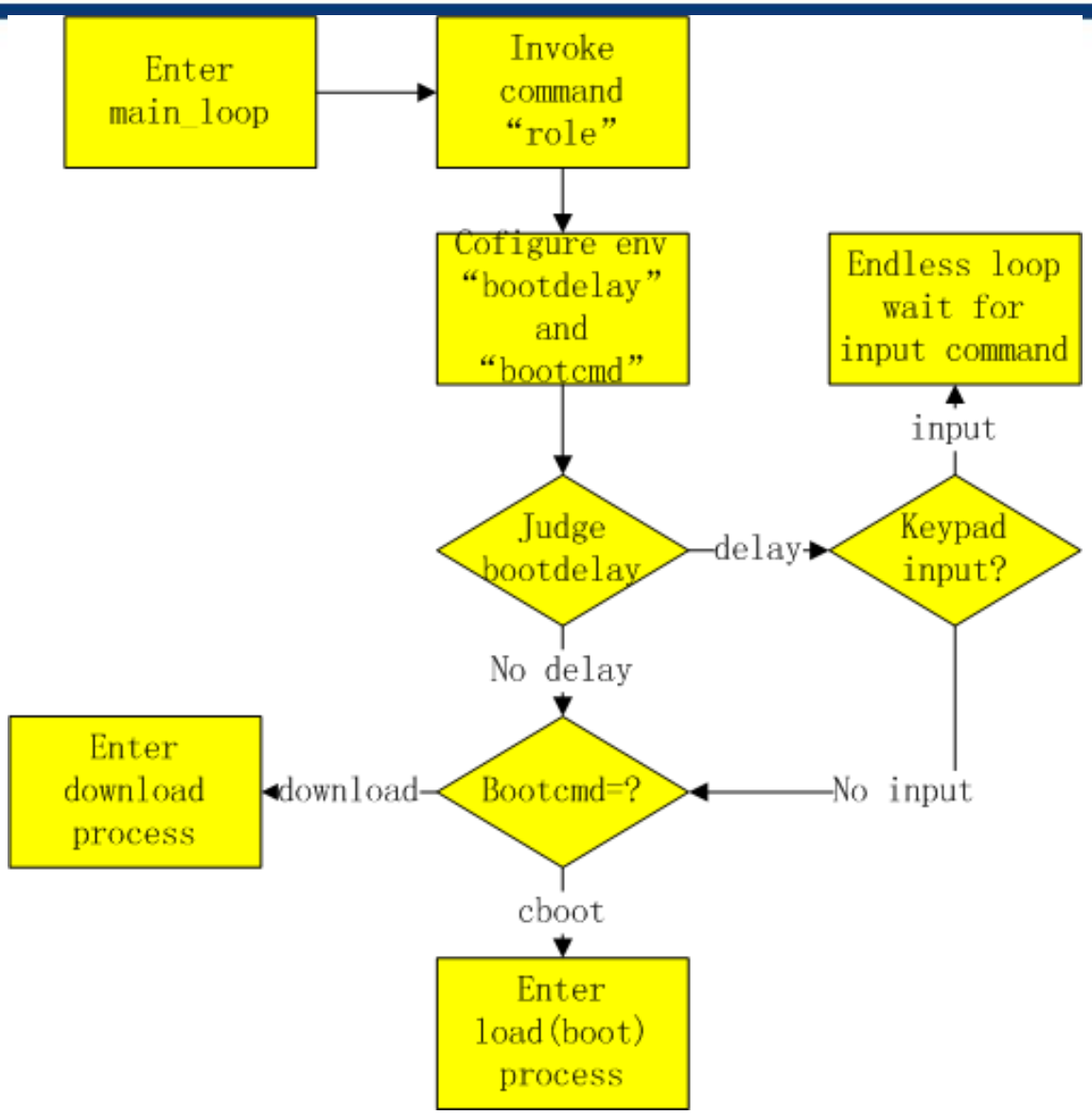
- Initialize all the drivers in order.



Initialization Process

Mode selection

- Chipram(fdl1 or spl) set a symble in internal RAM.
- Command “role” determines which mode will be entered in the next step via this symbol.



Initialization Process

Three important macros

- #define CONFIG_PREBOOT “role” : determines command “role” must be implemented before other command.
- #define CONFIG_BOOTDELAY 0 : 0 means do not accept commands from outside, just run the fixed process.
- #define CONFIG_BOOTCOMMAND “cboot normal” : means if no boot mode has been discovered by “role”, ”cboot normal”(i.e normal boot mode) will be adopted as default.

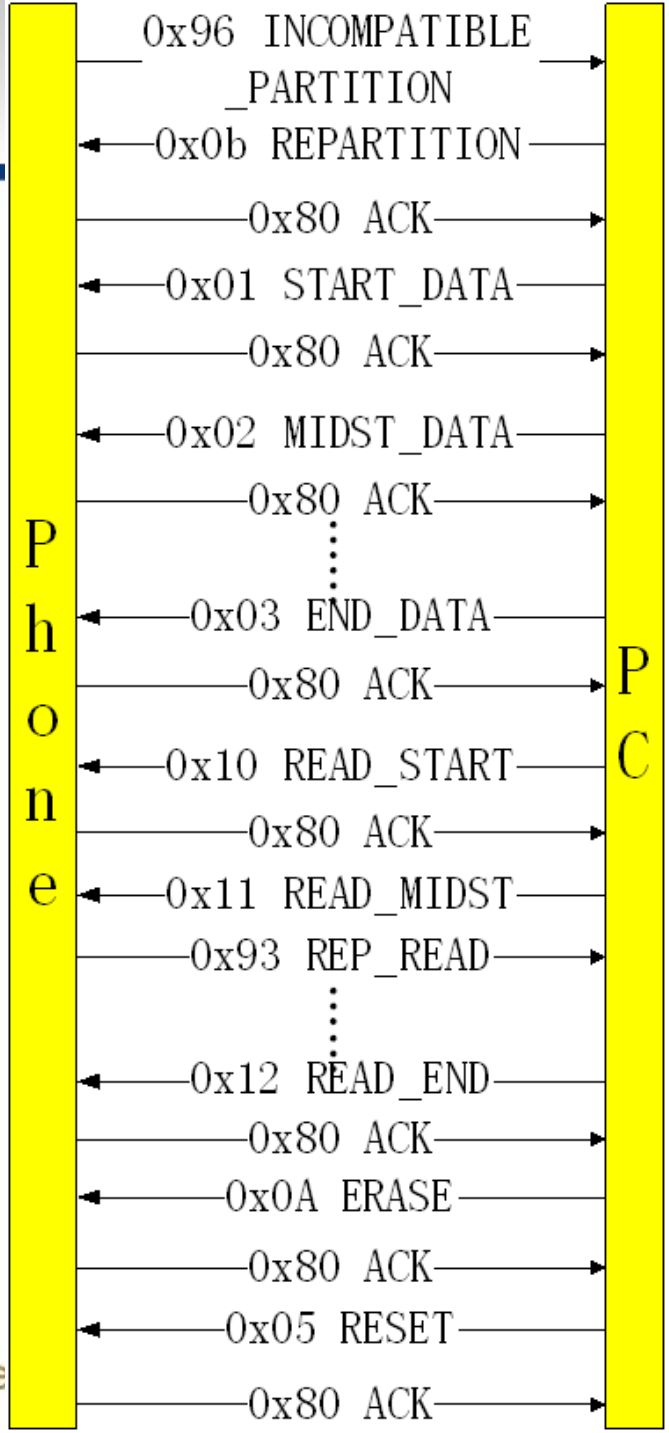
Agenda

- Overall Process
- Romecode
- 1st stage download FDL1
- 1st stage boot SPL
- U-boot Initialization Process
- U-boot Download Process
- U-boot Load Process

Download Process

A typical download process between Phone and PC

- PC(download tool) as a server
- Phone as a client
- Communicate with a simple protocol, whole process are controlled by PC.



Download Process

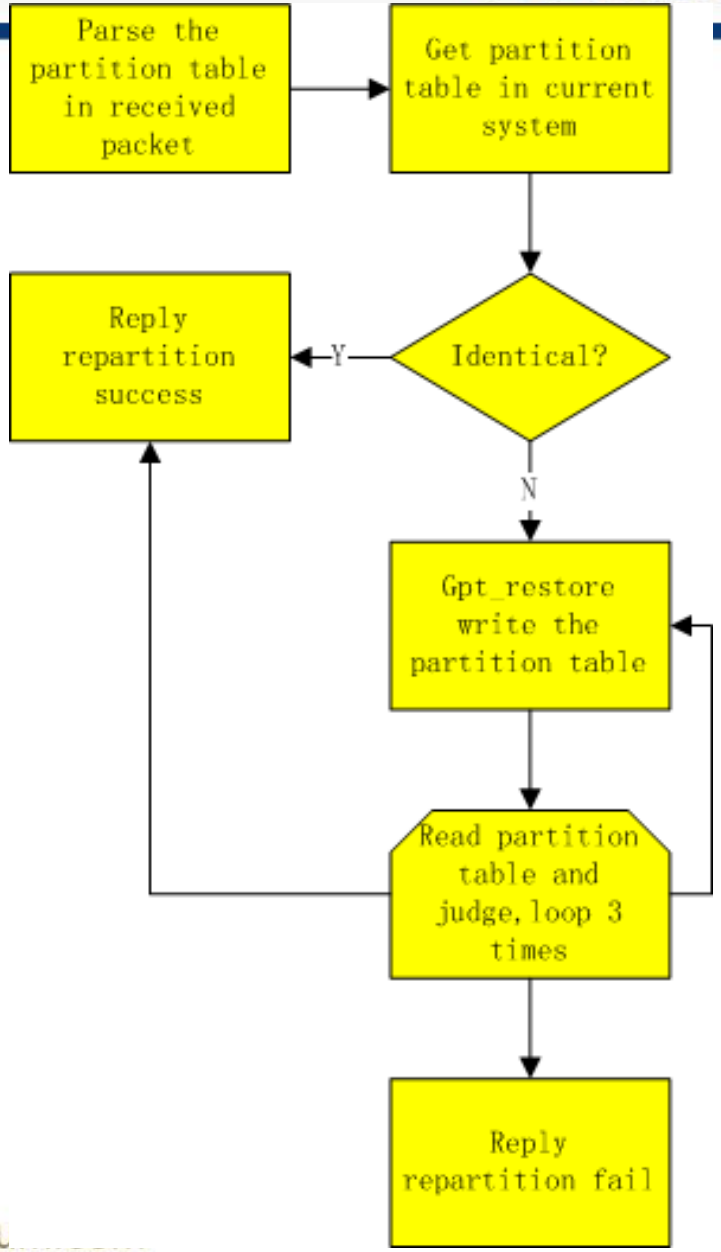
Four major functions:

- Repartition ： PC send 0x0B packet to request repartition , phone reply the repartition result.
- Download ： PC request begin with 0x01packet end with 0x03 packet, phone reply to each packet received.
- Read ： PC request begin with 0x10 packet end with 0x12 packet, phone reply to each packet received.
- Erase ： PC request erase operation by send 0x05 packet, phone reply the erase result.

Download Process

Repartition

- Repartition is always the first action of downloading a upgrade package.
- We use GPT format partition table.



Download Process

Download

- Key point of the u-boot
- Handle three types of packets send by PC:

  (1) 0x01START_DATA

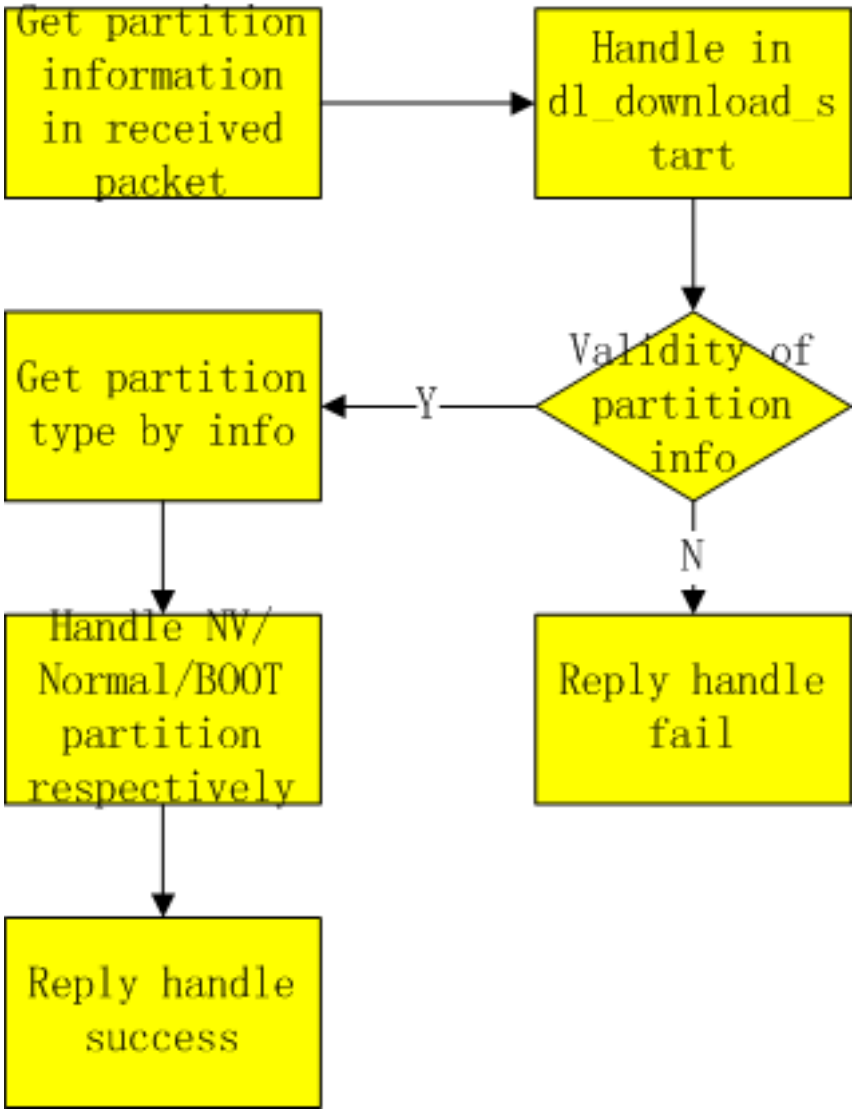
  (2) 0x02 MIDST_DATA

  (3) 0x03 END_DATA

Download Process

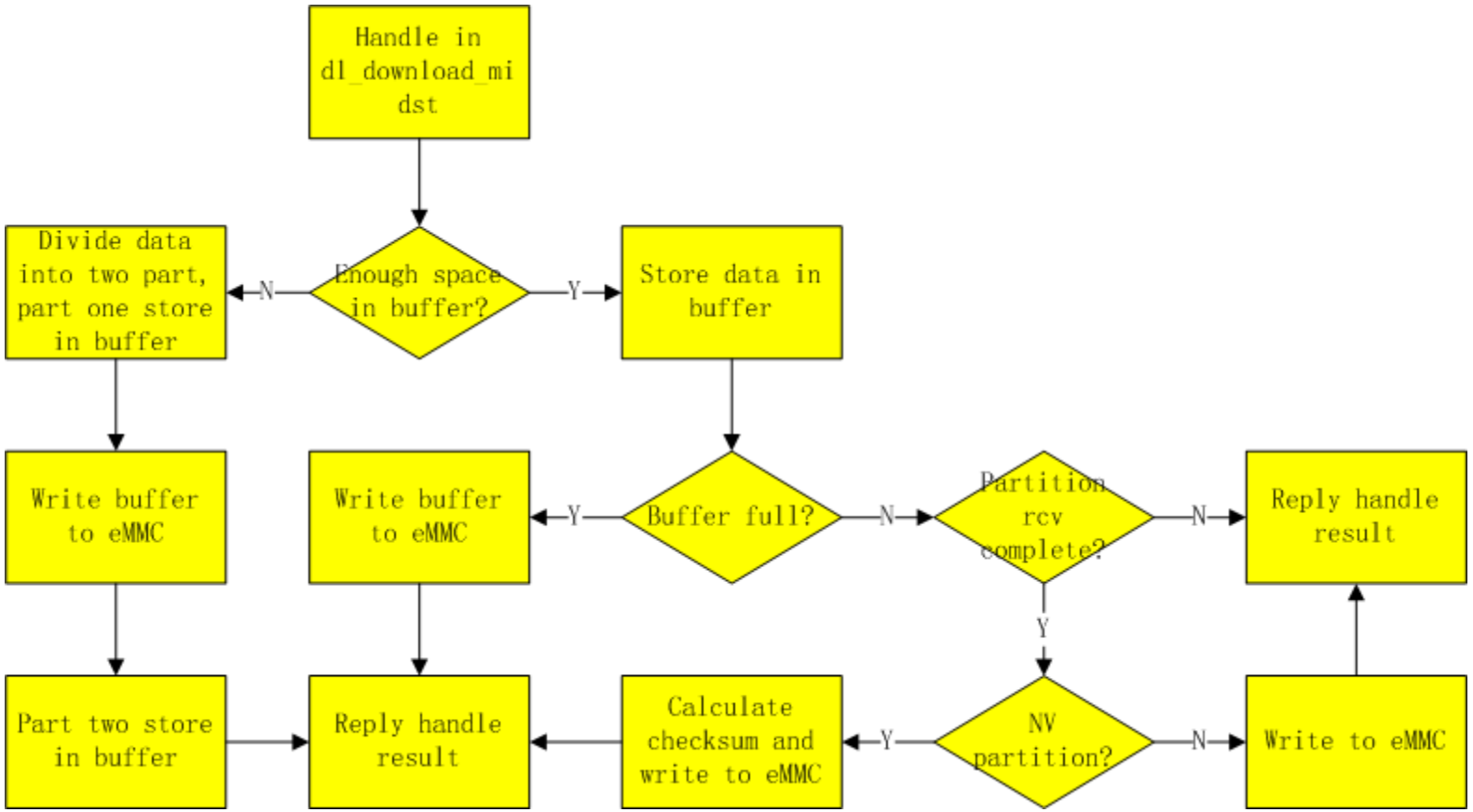
START_DATA procedure

- Obtain the information of the target partition and prepare the environment before the 0x02 MIDST_DATA packet arrive.



Download Process

MIDST_DATA procedure



Download Process

Special handling of MIDST_DATA

- Before write NV image to eMMC we must checksum the whole image to guarantee its accuracy.
- Sparse format images(USERDATA&CACHE) must be written into eMMC via a special API write_simg2emmc.
- Write spl and u-boot to their specific eMMC parts.

Download Process

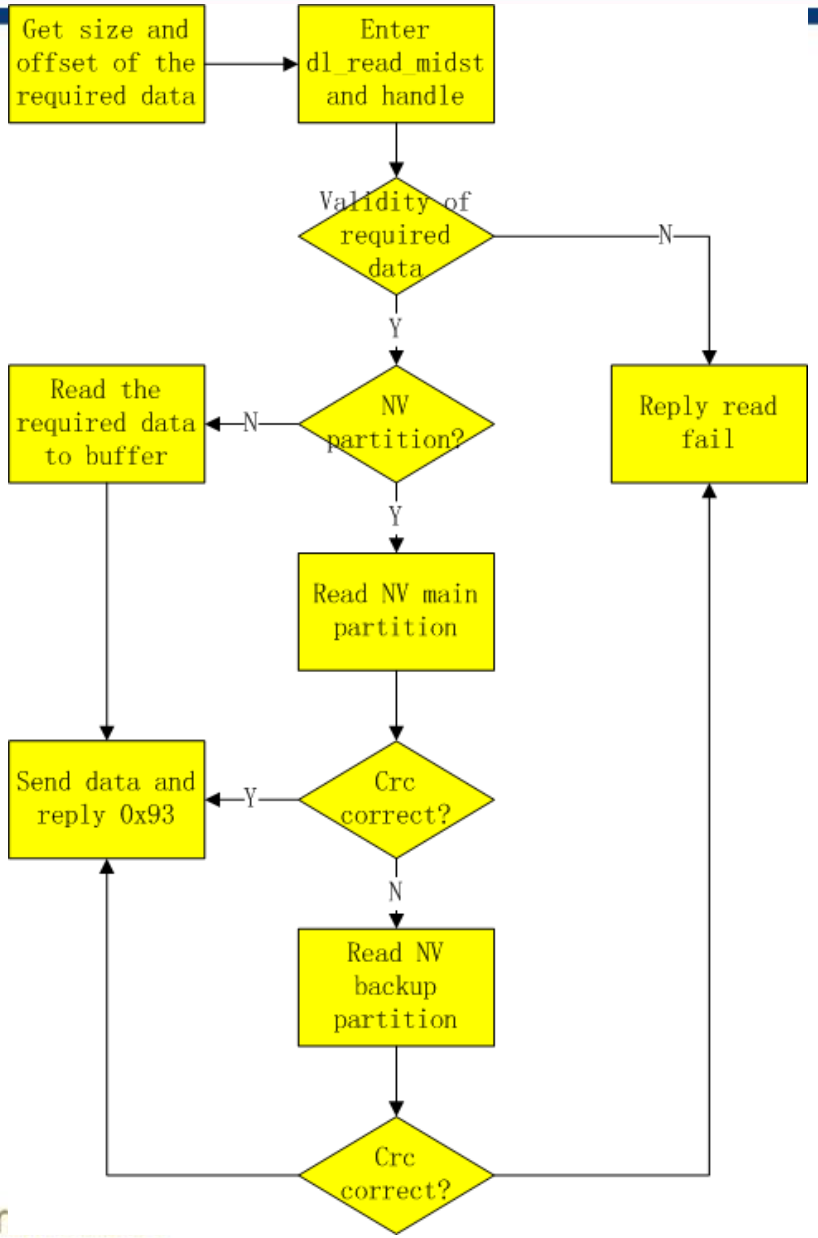
Read

- Two purposes of Read operation:
  (1) NV backup
  (2) System debug
- Handle three types of packets send by PC:
  (1) 0x10READ_START
  (2) 0x11 READ_MIDST
  (3) 0x12 READ_END

Download Process

READ_MIDST procedure

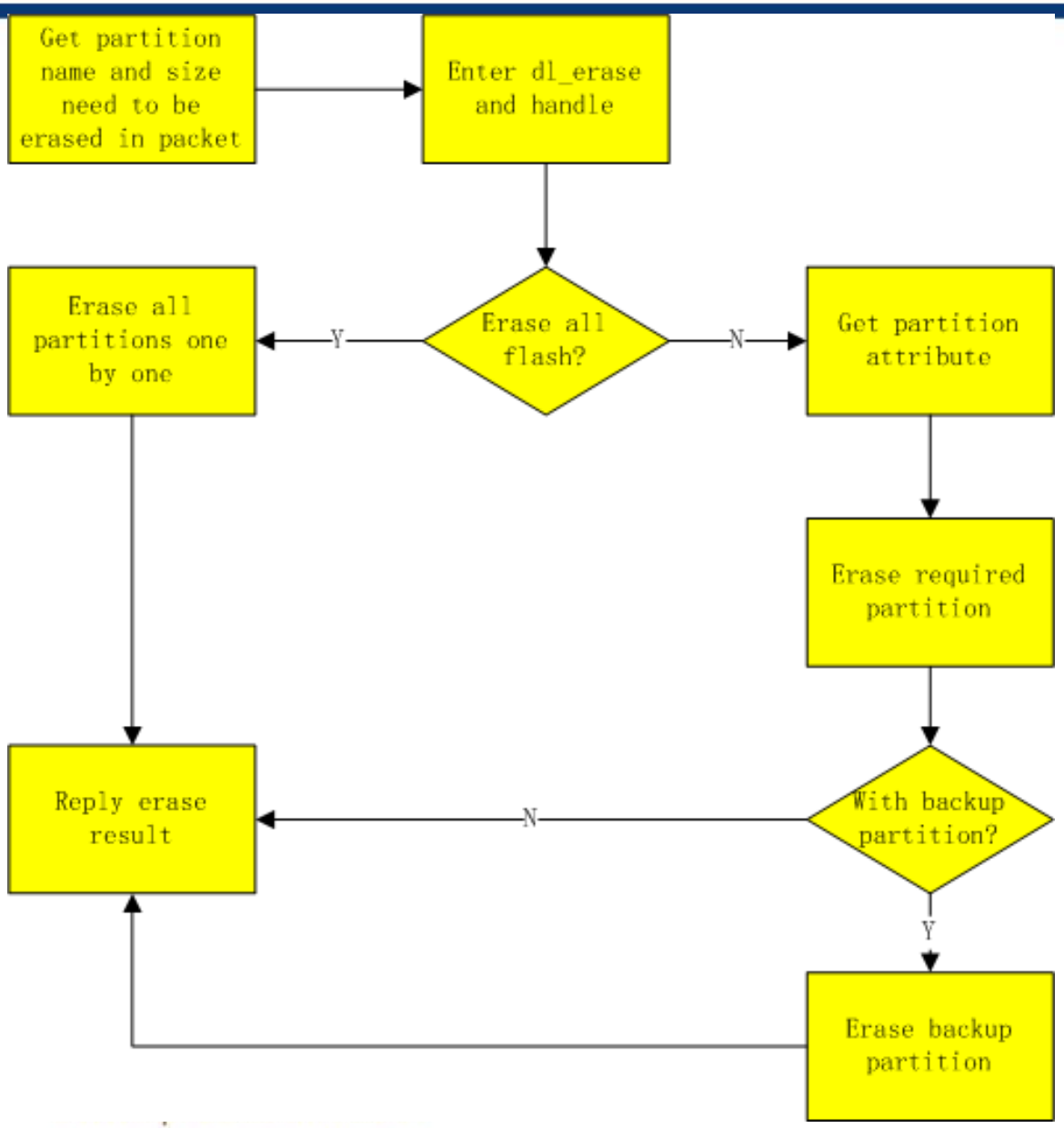
- Every read request packet contain required data's size and offset, so every interaction can be regarded as a independent procedure.
- All NV operations need crc verify.



Download Process

Erase

- Handle 0x0A ERASE packet
- Size=0xFFFFFFFF means erase all flash



Download Process

APIs provided by eMMC and USB driver

USB

- dl_packet_init : initialize packet list
- dl_get_packet : get packet from USB driver
- Dl_send_packet : send packet to USB driver

eMMC

- Emmc_write : write data to eMMC
- Emmc_read : read data from eMMC
- Emmc_erase : erase specified data in eMMC

Agenda

- Overall Process
- Romecode
- 1st stage download FDL1
- 1st stage boot SPL
- U-boot Initialization Process
- U-boot Download Process
- U-boot Load Process

Load Process

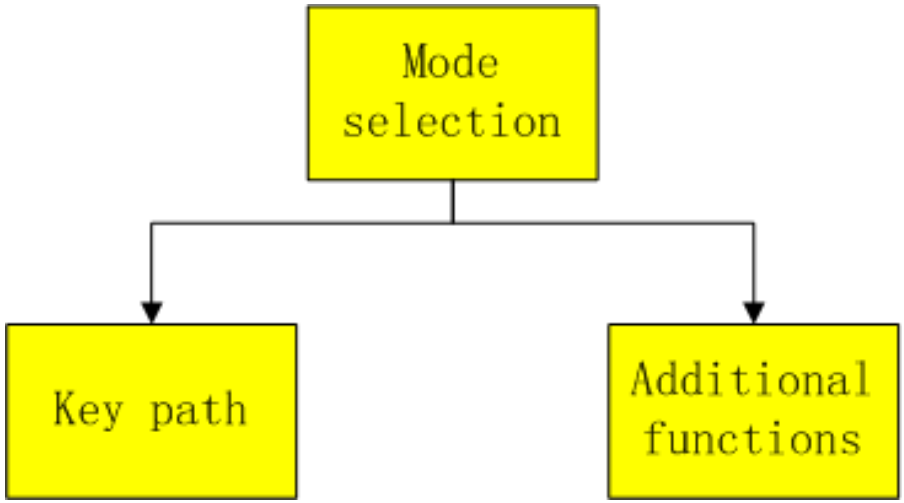
Look at a struct first:

```
typedef enum {
    CMD_UNDEFINED_MODE=0,
    CMD_POWER_DOWN_DEVICE,
    CMD_NORMAL_MODE,
    CMD_RECOVERY_MODE,
    CMD_FASTBOOT_MODE,
    CMD_ALARM_MODE,
    CMD_CHARGE_MODE,
    CMD_ENGTEST_MODE,
    CMD_WATCHDOG_REBOOT ,
    CMD_SPECIAL_MODE,
    CMD_UNKNOW_REBOOT_MODE,
    CMD_PANIC_REBOOT,
    CMD_CALIBRATION_MODE,
    CMD_AUTODLOADER_REBOOT,
    CMD_AUTOTEST_MODE,
    CMD_EXT_RSTN_REBOOT_MODE,
    CMD_IQ_REBOOT_MODE,
    CMD_MAX_MODE
}boot_mode_enum_type;
```

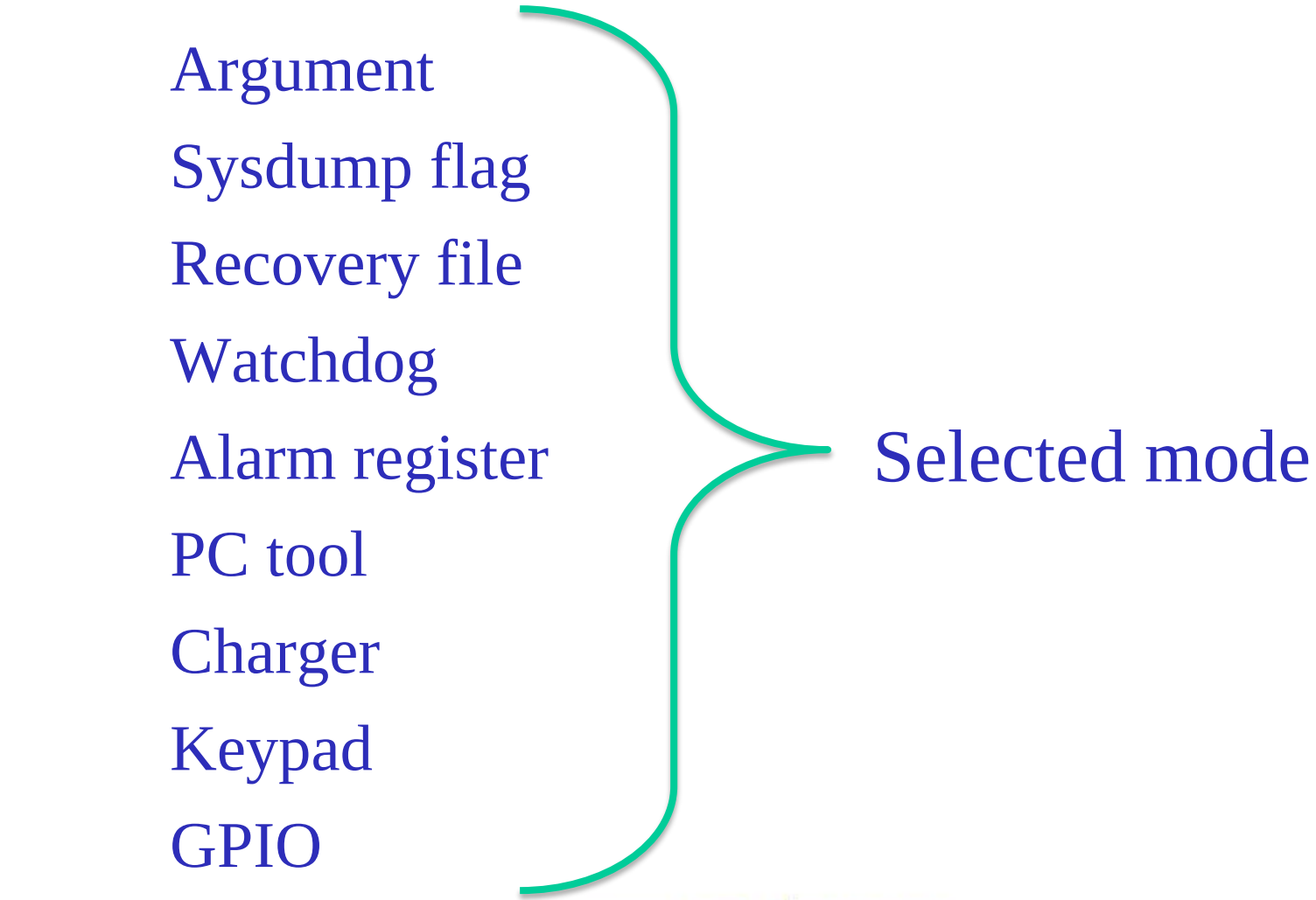
Load Process

Three parts:

- Mode selection : find out which mode need to enter in many ways.
- Key path : most important works u-boot supposed to do.
- Additional functions : other works accomplished by u-boot.



Load Process



Load Process

Mode selection details(1)

- Argument(get_mode_from_arg) : enter command “cboot xxx” will select xxx mode, e.g “cboot charge”. Mostly for test and debug.
- Sysdump flag(write_sysdump_before_boot_extend) : check flag written by kernel and determine whether dump RAM content to SD card.
- Recovery file(get_mode_from_file_extend) : check file in the misc partition and determine whether load the recovery image instead of boot image.

Load Process

Mode selection details(2)

- Watchdog(get_mode_from_watchdog) : check the watchdog register and determine which mode will be entered in.
- Alarm register(get_mode_from_alarm_register) : check a set of registers and files to determine whether enter into alarm mode and whether it's the right time to wake up the phone.
- PC tool(get_mode_from_pctool) : communicate with PC tools(e.g MobileTester) and determine whether enter into the calibration mode or autotest mode.

Load Process

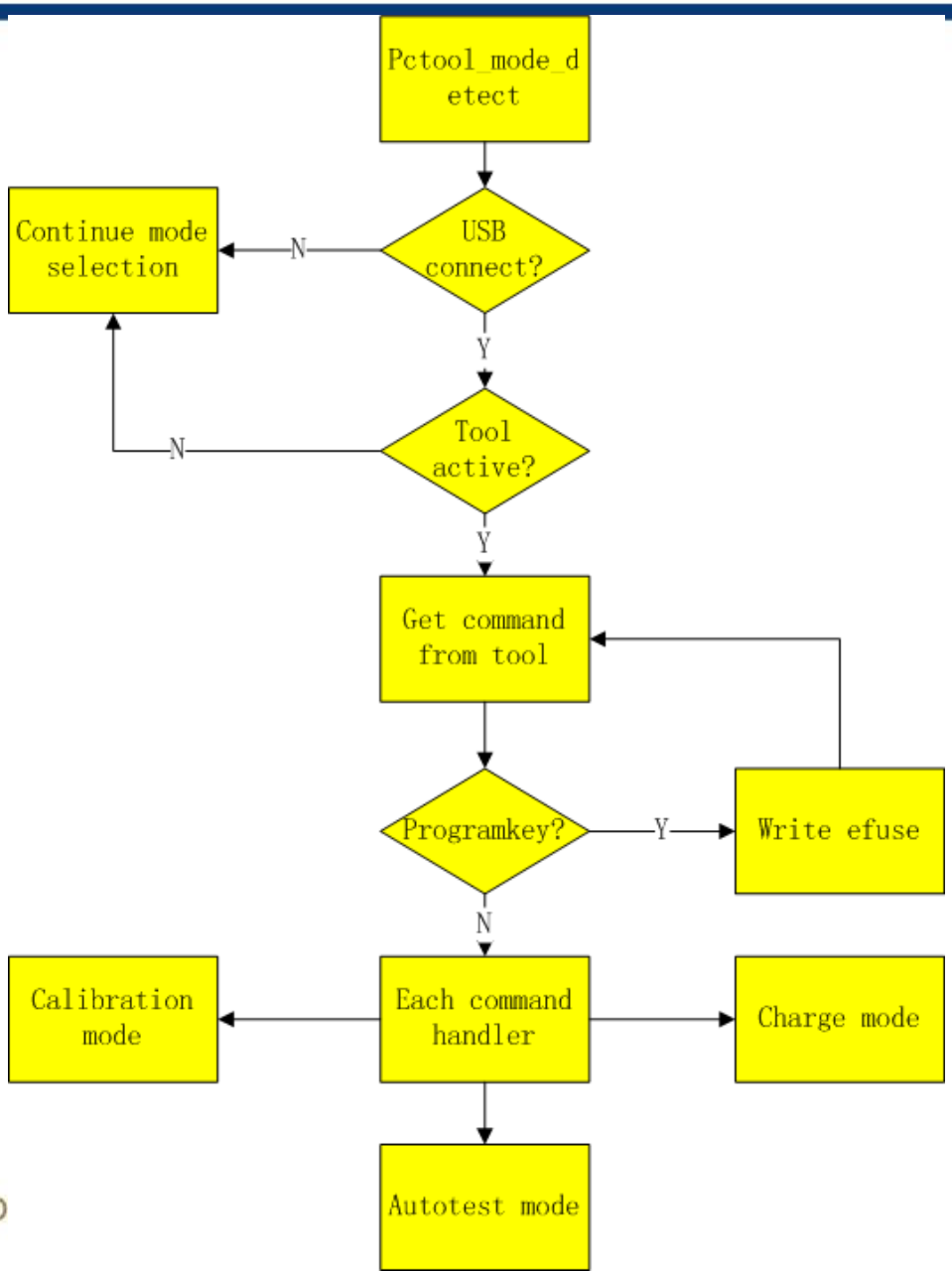
Mode selection details(3)

- Charger(get_mode_from_charger) : judge whether charger has been connected and whether enter into the charge mode.
- Keypad(get_mode_from_keypad) : monitor the keypad operation then determine the boot mode, support three specific operations at present.
- GPIO(get_mode_from_gpio_extend) : an optional method, check a specific GPIO register and determine the boot mode.

Load Process

Get mode from PC tools

- If u-boot find that USB has been connected, it will wait for command from PC tool, these command determines which boot mode will be entered in.
- Efuse programe also be accomplished here.



Load Process

Key path

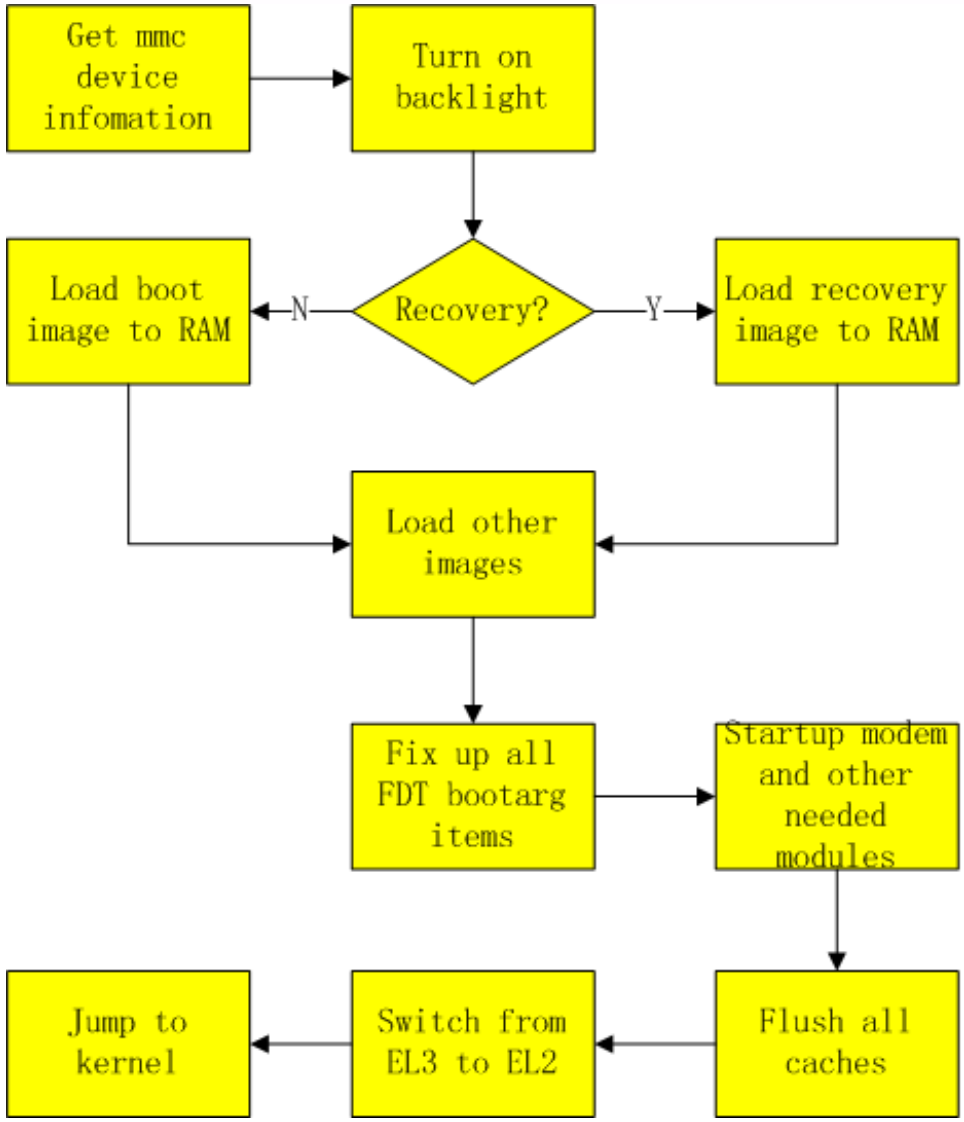
Four missions

- Load boot image(in recovery mode is recovery image).
- Startup Modem and other independent modules.
- Fix up FDT bootargs.
- Jump to kernel.

Load Process

Key path

- "other images" include NV images, modem&DSP images, power management system image.
- Before jump to kernel we need to startup the modem and power management system.
- Must switch to EL2 in armv8 architecture.



Load Process

Load boot(recovery) image

- Parse the header in boot image and get the layout of the next sections.
- Load kernel to appropriate location.
- Load ramdisk.
- Check and load DTB.

Boot image formation

Load Process

展讯通信 SPREADTRUM

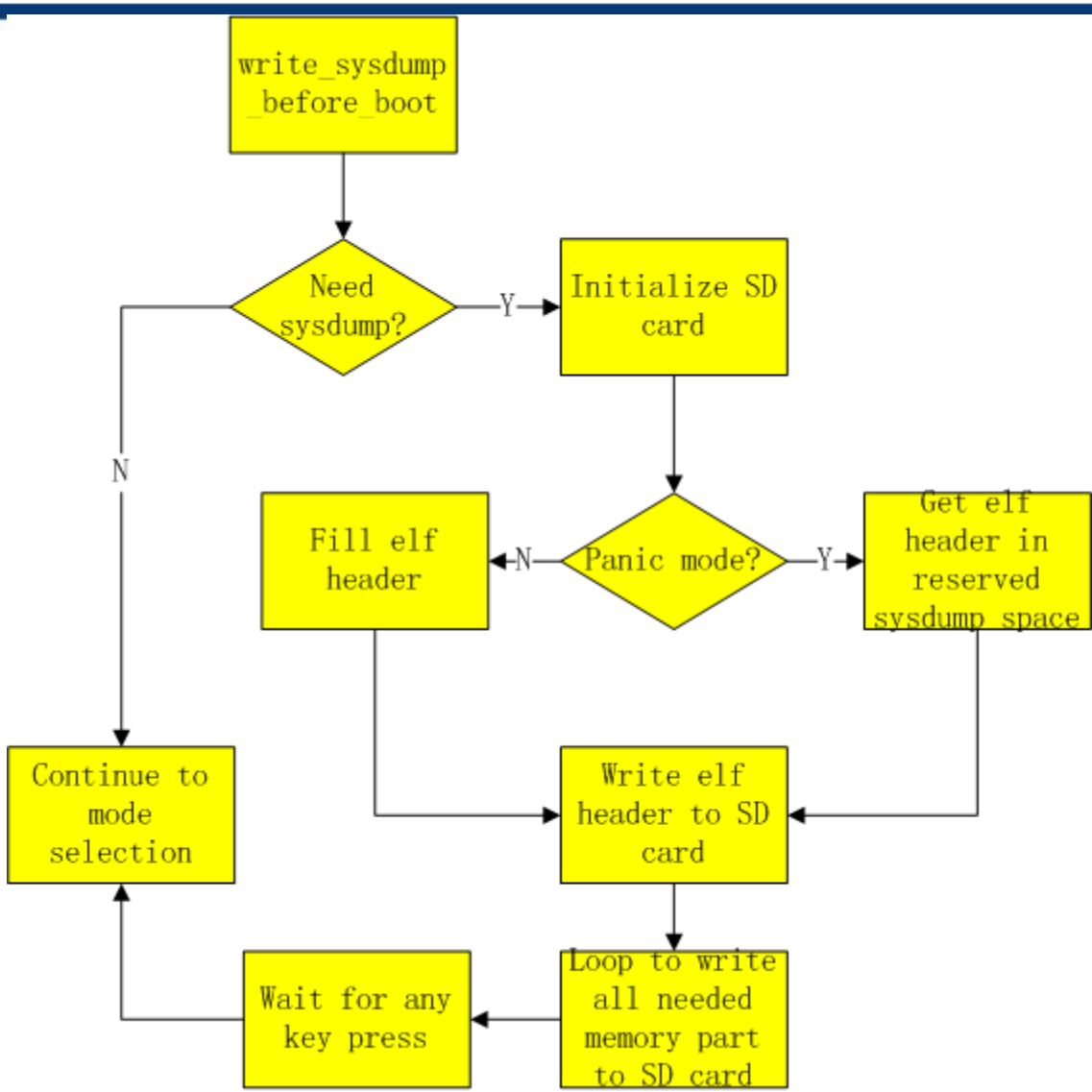
Additional functions

- Sysdump : dump all RAM or part of RAM required by kernel, for debug use.
- Fastboot : a standard upgrade method in android system.
- Autodloader : automatic download the whole upgrade package in auto test environment.

Load Process

Sysdump

- First Check the watchdog register, watchdog /unknown/rstn/special reboot mode will cause the dump action.
- Dump procedure will display on LCD panel.



Load Process

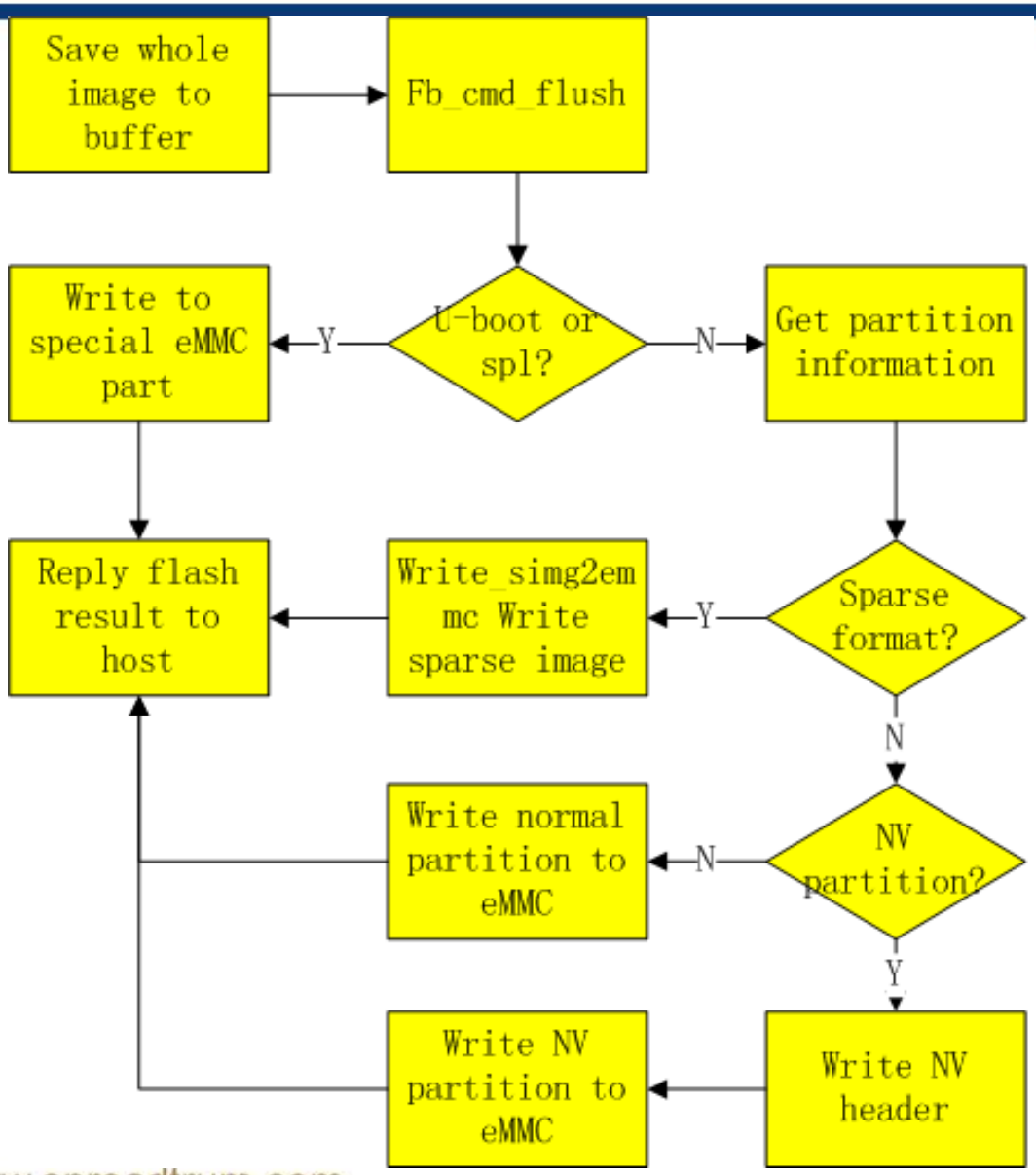
Fastboot

- Similar to download process.
- Communicate with fastboot host via USB.
- Support command flash(download), erase, reboot, getvar at present.
- Flash is the most important command of fastboot.
- Use “adb reboot bootloader” to get into the fastboot mode.

Load Process

Fastboot

- Similar to download process, normal/NV/sparse image will be dealed with different way.



Load Process

Autodloader

- Used for download package automatically in lab or factory.
- When u-boot enter the autodloader mode, it will replace the role of RomCode and fdl1 temporarily.
- Download fdl1 and fdl2(u-boot) from ResearchDownload tool and discard them, then jump to download process to download all rest images in upgrade package.

Appendix

New features in u-boot64 compare to u-boot32

- Merge fdl2 and u-boot into one binary file.
- Some modules use original u-boot code instead of spreadtrum code, include EXT4, EFI partition, MMU.
- Support command input in console.
- Reconsitution all the Spreadtrum code , remove the “property” directory then add “common/loader” and “common/dloader” directories, many files and functions have been renamed or relocated.

Appendix

New features in u-boot64 compare to u-boot32(2)

- Remove all the old SoC support.
- Remove all the unnecessary spreadtrum code in initialization process.
- Remove all the redundant driver code and include files in arch/arm directory.
- Modifications for armv8 support.
- Rewrite part of the fastboot and download code.
- Redesign the autodloader function.

Thanks!