U-boot

Bootload

- What is bootloader
- Bootloader is the first execute program on SOC (?)
- Boot : short bootstrap
  - Initialize basic of SOC (CPU, RAM, CLK)
- Down loader
  - Download Image or Application to ram from host (developer host PC)
- Loader
  - Load OS to ram form volatile memory

All kinds of embedded Linux bootloader

- U-boot
- UEFI
- Redboot
- Stubby (Linaro) …
- Anyway, they are same target
  - Load and boot OS to RAM from storage

Concepts of the Boot Loader

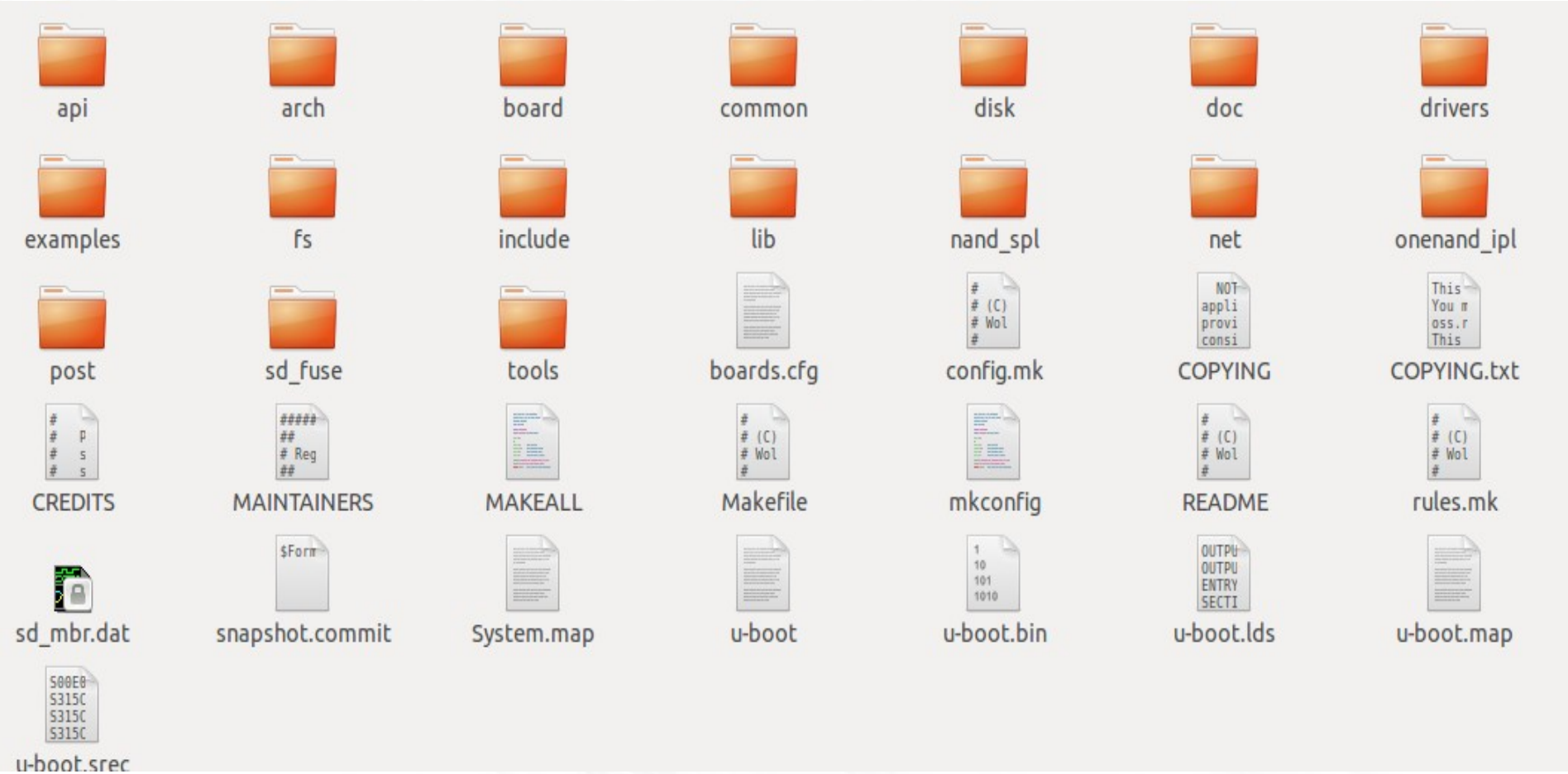
- Boot Loader is varied from CPU to CPU,from board to board.
- All the system software and data are stored in some kind of nonvolatile memory.
- Operation Mode of Boot Loader
  - Boot : Initialize basic of SOC
  - Load : load OS to RAM

Introduce U-boot

U-boot

- Das U-Boot -- the Universal Boot Loader
- http://www.denx.de/wiki/U-Boot
- GitHub for u-boot
- Open Source follow GPL
- Supply many CPU
  - PPC, ARM, x86, MIPS, AVR32 …
- Supply basic periphery devices
  - UART, Flash, SD/MMC ….

u-boot directory structure



u-boot directory structure

- Arch
  - Many types CPU : Arm, mips, i386 ...
- Board
  - Many types develop board : Samsung, ti, davinci ...
- Tools
  - Make Image (u-boot, linux) or S-Record image tool
- Drivers
  - Some HW control code

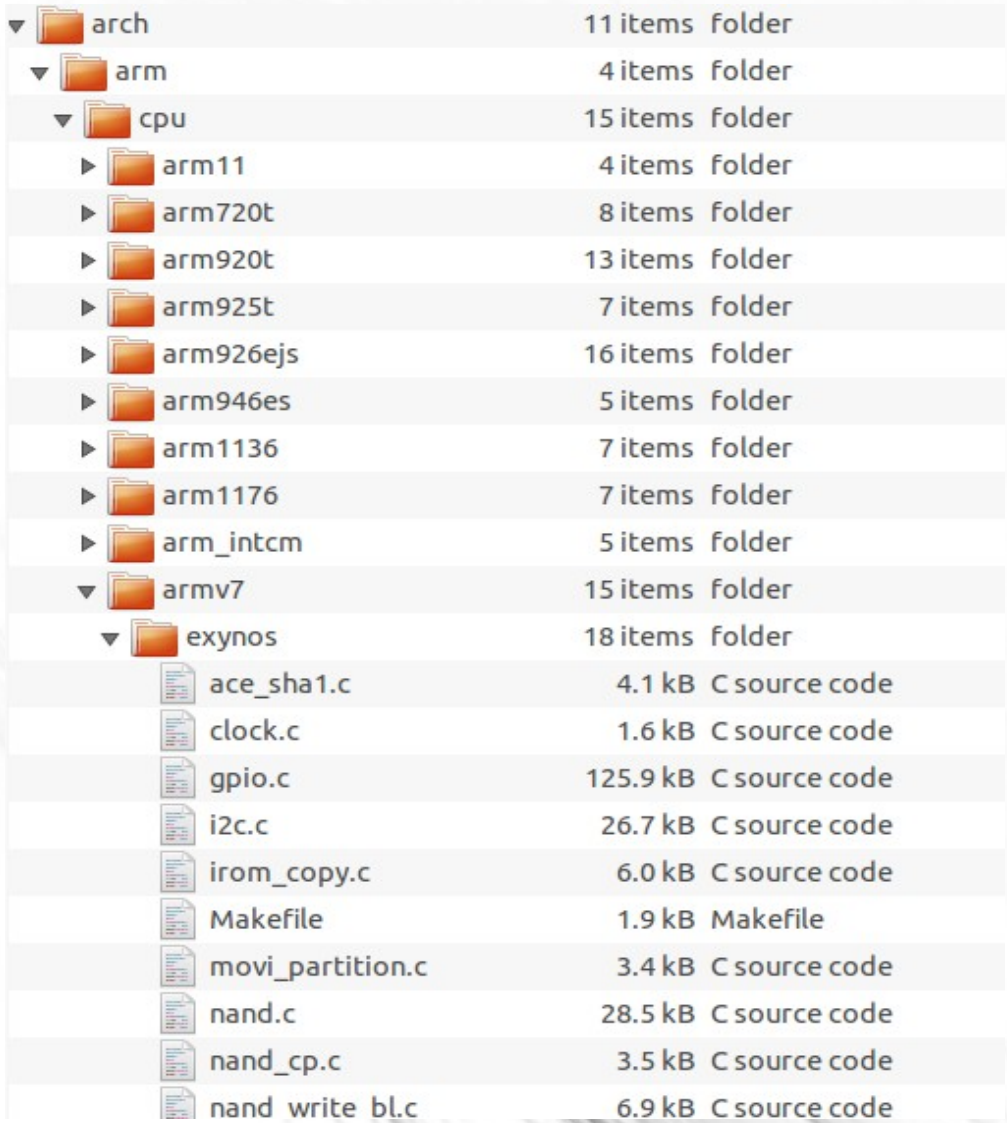
u-boot directory structure

- Common
  - Major command and relation environment setting source code
- Api
  - Implement unrelated hardware code
- nand_spl, onenand_ipl
  - Related nand/onenand flash control
- Example
  - Standalone application

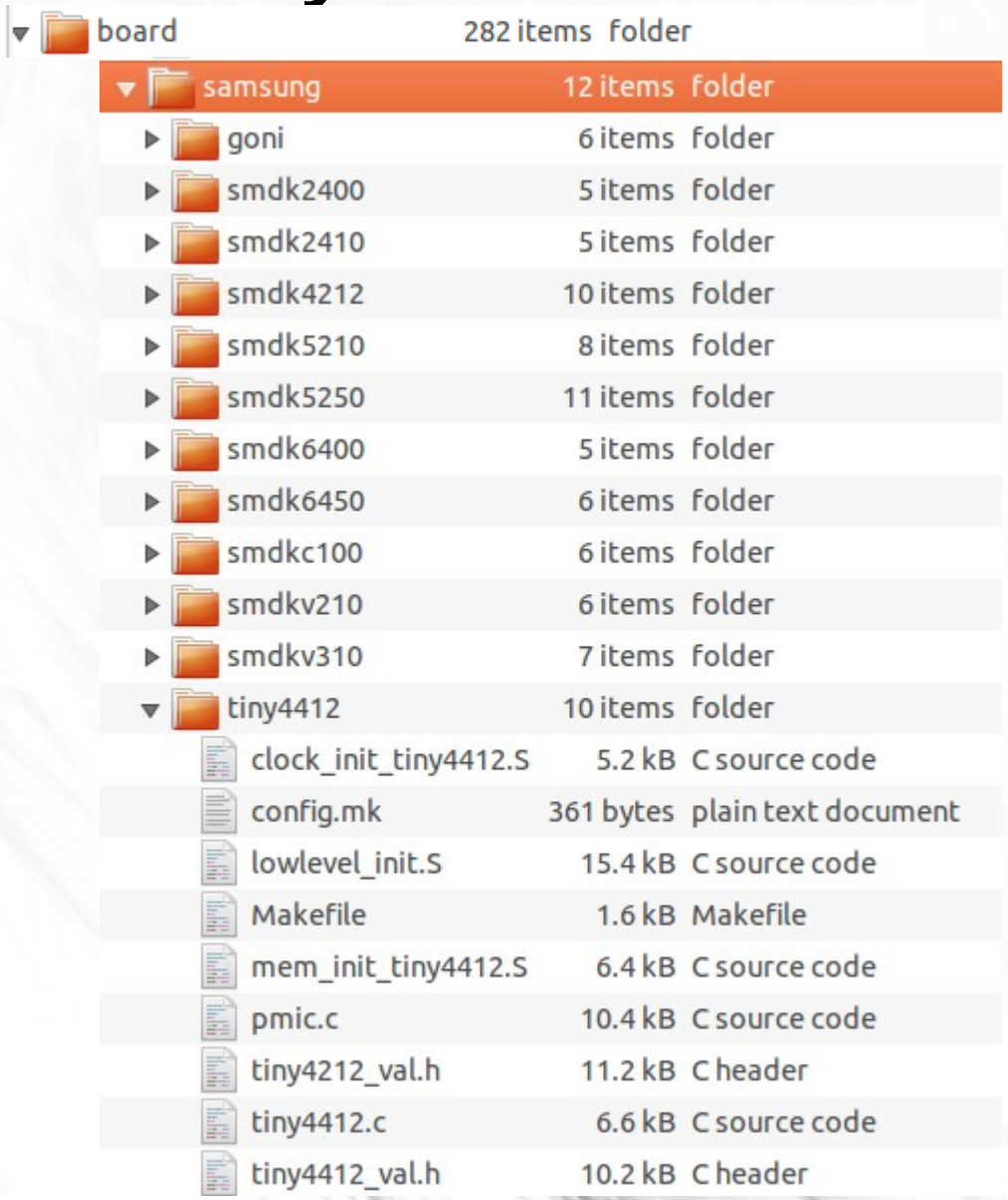
u-boot directory structure

- Post
  - Supply Power On Self Test function
- Fs
  - Supply file system : fat, jffs2, ext2, cramfs
- Lib
  - General public library : CRC32/16, bzlib, ldiv ..
- Disk
  - Supply disk driver and partition handling

u-boot directory structure about tiny4412



u-boot directory structure about tiny4412



u-boot directory structure about tiny4412

- **arch/arm/cpu/armv7/exynos/**
  - Samsung exynos CPU related
  - Clock, i2c, irom, mmc, emmc ...
- **board/samsung/tiny4412/**
  - Tiny4412 EVB related
  - Low level init, memory init, link script ...
- Common
  - Tiny4412 u-boot command related
- **include/configs/tiny4412.h**
  - Tiny4412 EVB build configure related

How to build u-boot

- Clear
  - **#make distclean**
- Configure
  - **#make board_config <tiny4412_config>**
- Build
  - **#make -j4**
- Result of build
  - u-boot : ELF format file
  - **u-boot.bin** : raw data binary

Link Script

- u-boot.lds
  - board/samsung/tiny4412/u-boot.lds
  - The link script will pack all into binary.
  - The binary file will put in storage.
  - The start address (.TEXT) can be modified.
- CONFIG_SYS_TEXT_BASE
  - board/samsung/tiny4412/config.mk
  - Check u-boot.map

u-boot.map

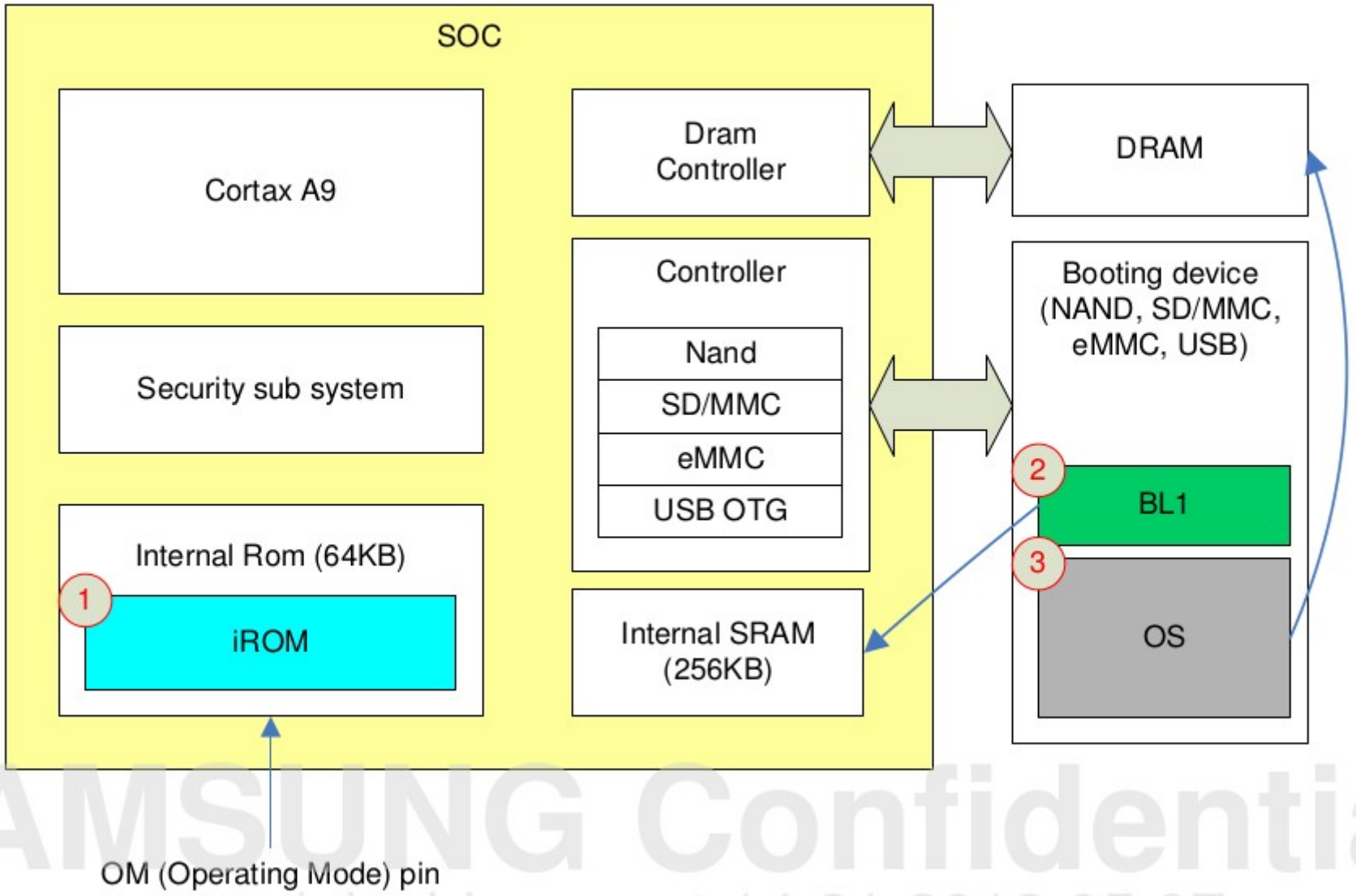
```
                0x00000000                . = 0x0
                0x00000000                . = ALIGN (0x4)

.text           0xc3e00000     0x2d6e8
 arch/arm/cpu/armv7/start.o(.text)
 .text          0xc3e00000      0x460 arch/arm/cpu/armv7/start.o
                0xc3e00000                _start
                0xc3e00040                _end_vect
                0xc3e00040                _TEXT_BASE
                0xc3e00044                _bss_start_ofs
                0xc3e00048                _bss_end_ofs
                0xc3e0004c                IRQ_STACK_START_IN
                0xc3e00074                relocate_code
 board/samsung/tiny4412/libtiny4412.o(.text)
 .text          0xc3e00460      0x154c board/samsung/tiny4412/libtiny4412.o
                0xc3e00464                cache_init
                0xc3e00468                lowlevel_init
                0xc3e0064c                uart_asm_init
                0xc3e0085c                nand_asm_init
                0xc3e00930                theLastJump
                0xc3e00a30                mem_ctrl_asm_init
                0xc3e00d80                system_clock_init
                0xc3e00fec                board_init
                0xc3e010f4                dram_init
                0xc3e010fc                dram_init_banksize
                0xc3e01174                board_eth_init
                0xc3e0117c                checkboard
                0xc3e0119c                board_late_init
```

Link Script

arch/arm/cpu/armv7

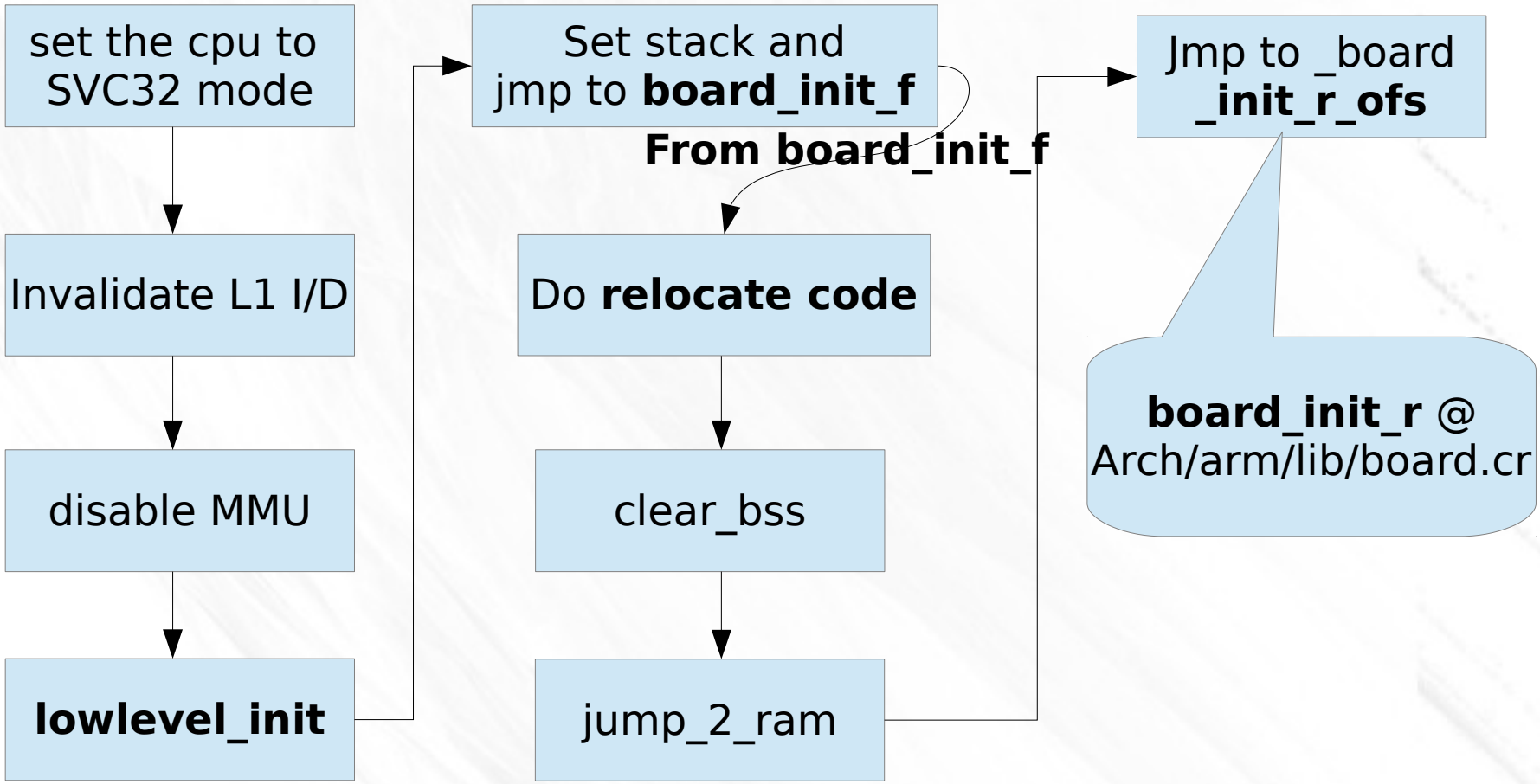
```
OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
OUTPUT_ARCH(arm)
ENTRY(_start)
SECTIONS
{
	. = 0x00000000;

	. = ALIGN(4);
	.text :
	{
		arch/arm/cpu/armv7/start.o (.text)
		*(.text)
	}

	. = ALIGN(4);
	.rodata : { *(SORT_BY_ALIGNMENT(SORT_BY_NAME(.rodata*))) }

	. = ALIGN(4);
	.data : {
		*(.data)
	}

	. = ALIGN(4);

	. = .;
	__u_boot_cmd_start = .;
	.u_boot_cmd : { *(.u_boot_cmd) }
	__u_boot_cmd_end = .;

	. = ALIGN(4);

	.rel.dyn : {
		__rel_dyn_start = .;
		*(.rel*)
		__rel_dyn_end = .;
	}

	.dynsym : {
		__dynsym_start = .;
		*(.dynsym)
	}
```

Exynos 4412 Booting Sequence

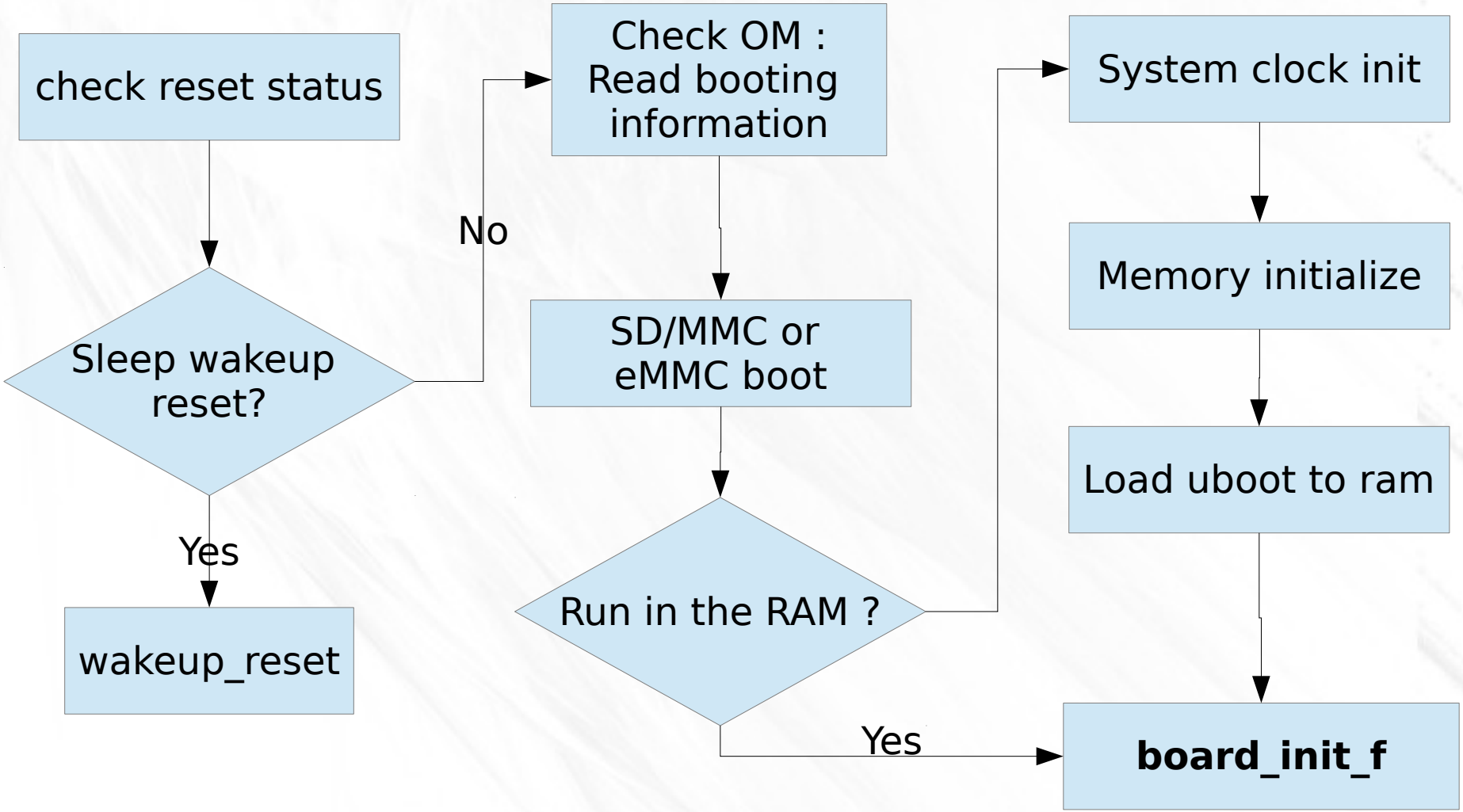


U-boot initialize sequence

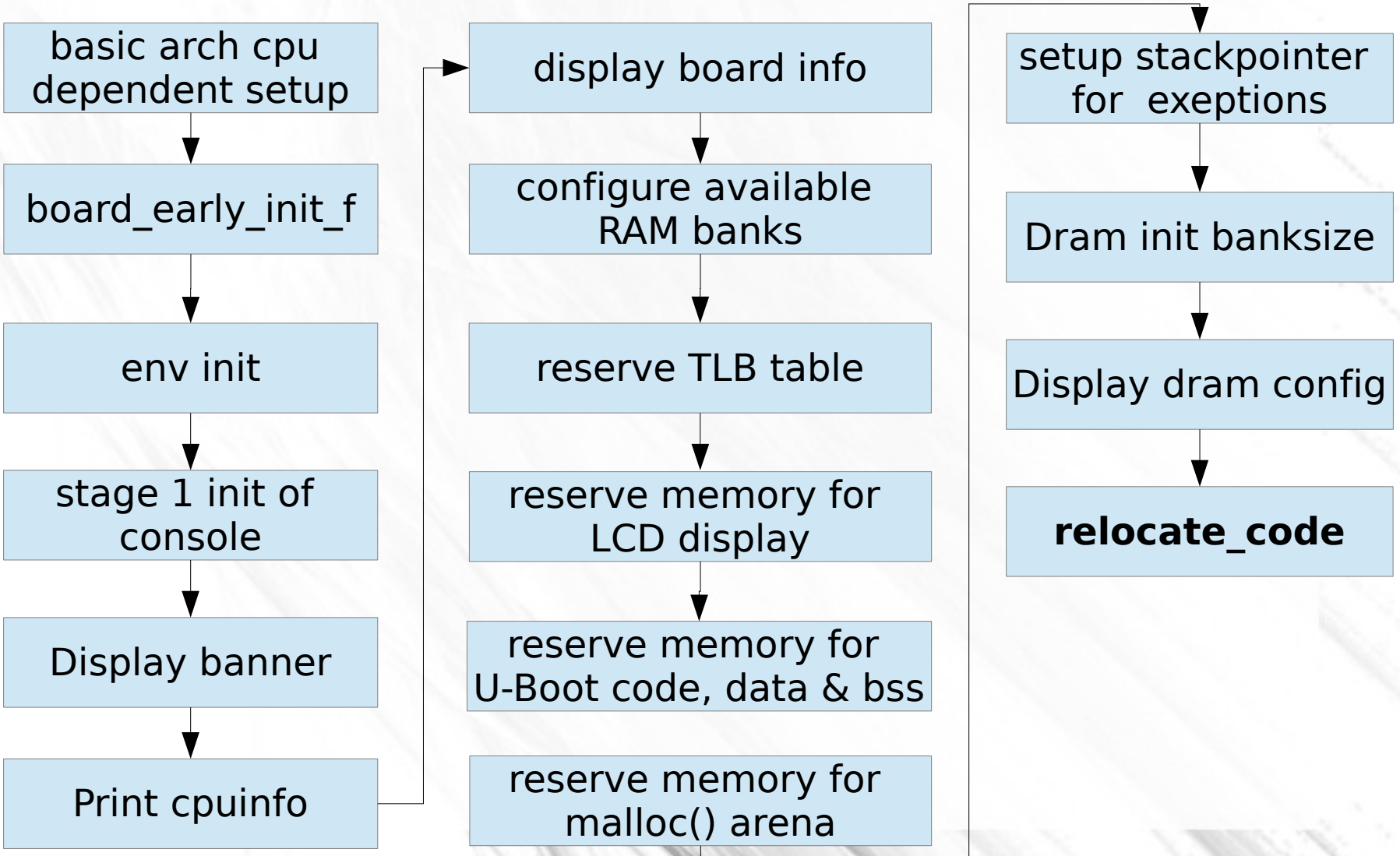
U-boot start up sequence



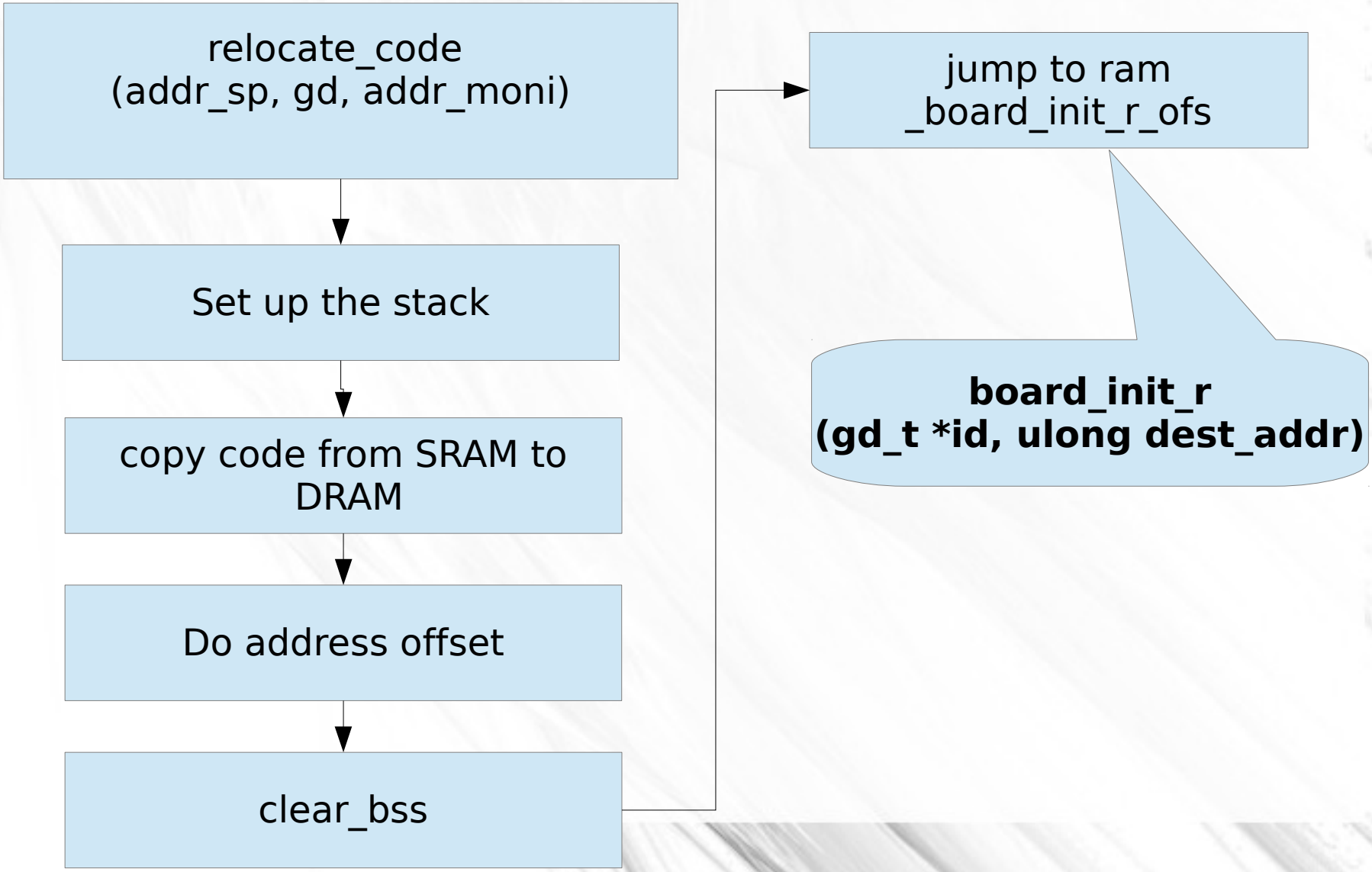
lowlevel_init



board_init_f


basic arch cpu dependent setup
board_early_init_f
env init
stage 1 init of console
Display banner
Print cpuinfo
display board info
configure available RAM banks
reserve TLB table
reserve memory for LCD display
reserve memory for U-Boot code, data & bss
reserve memory for malloc() arena
setup stackpointer for exeptions
Dram init banksize
Display dram config
relocate_code

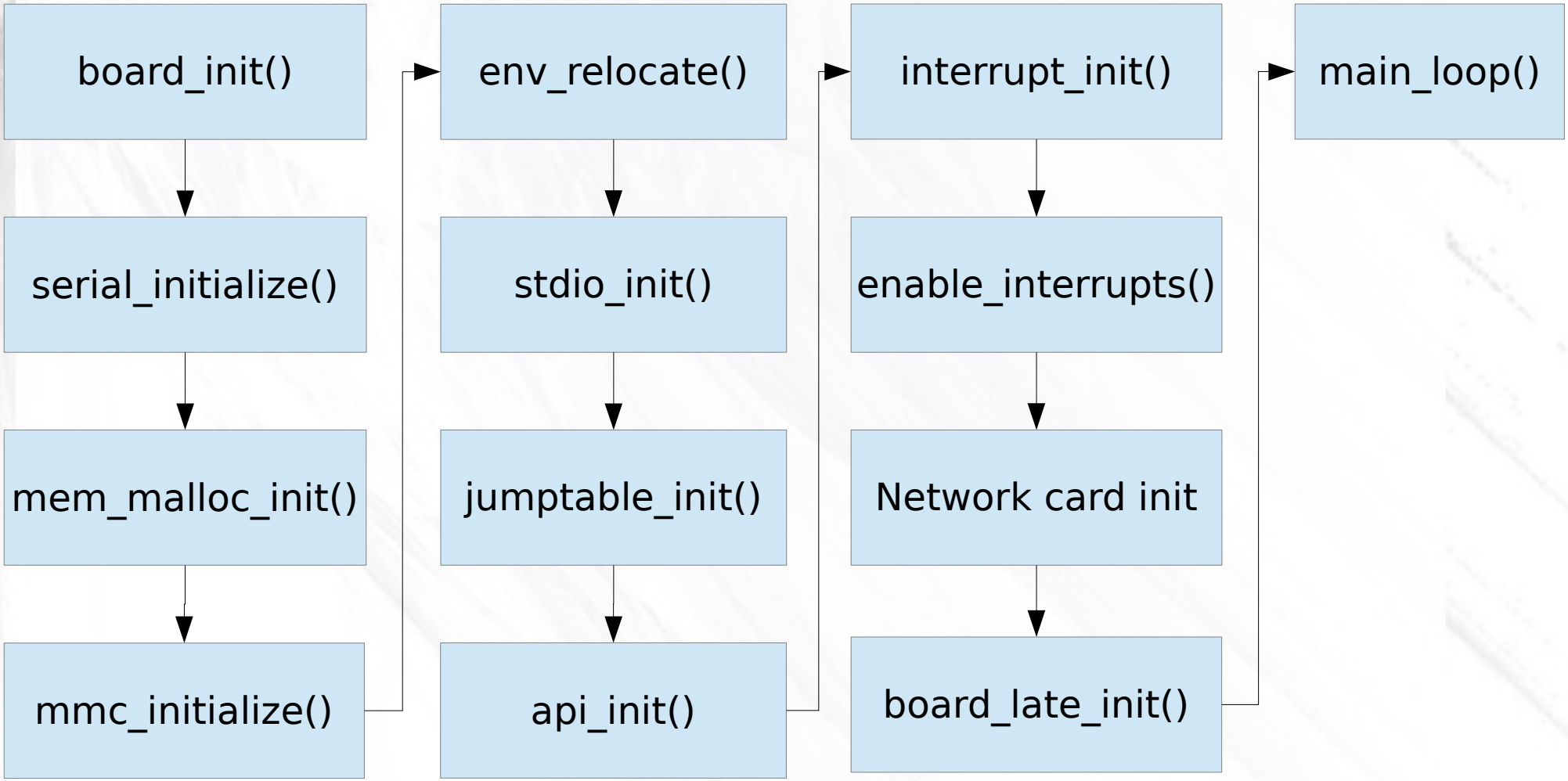
relocate code



board_init_r

```
/************************************************************************
 *
 * This is the next part if the initialization sequence: we are now
 * running from RAM and have a "normal" C environment, i. e.
 * global data can be written, BSS has been cleared, the stack size in
 * not that critical any more, etc.
 *
 ************************************************************************
 */
```

board_init_r


board_init()
serial_initialize()
mem_malloc_init()
mmc_initialize()
env_relocate()
stdio_init()
jumptable_init()
api_init()
interrupt_init()
enable_interrupts()
Network card init
board_late_init()
main_loop()

Operating U-boot

Operating U-boot

- Understand and use command
- Understand and modify parameters
- Run Application

Add Feature

- Add command
  - common/
- Add driver
  - driver
- Add application
  - example
- boards.cfg

Add Command

- How to create a command ?
- Directory
  - common/
    - cmd_mmc.c, cmd_bootm.c, cmd_help.c
- U_BOOT_CMD(name,maxargs,rep,cmd,usage,help)
  - include/command.h

Add application

- How to add a application
- Directory
  - example
- Api
  - Standalone
  - arch/arm/config.mk
    - STANDALONE_LOAD_ADDR

Add a driver

- How to add a driver
- Directory
  - driver/
  - drivers/gpio/s5p_gpio.c

Boot Linux kernel

Boot Linux kernel

- Boot command
  - Bootm
  - common/cmd_boot.c
- uImage
  - Tool : mkimage
- Linux kernel ATAG

uImage

- tools/mkimage

```
Usage: tools/mkimage -l image
        -l ==> list image header information
      tools/mkimage [-x] -A arch -O os -T type -C comp -a addr -e
ep -n name -d data_file[:data_file...] image
        -A ==> set architecture to 'arch'
        -O ==> set operating system to 'os'
        -T ==> set image type to 'type'
        -C ==> set compression type 'comp'
        -a ==> set load address to 'addr' (hex)
        -e ==> set entry point to 'ep' (hex)
        -n ==> set image name to 'name'
        -d ==> use image data from 'datafile'
        -x ==> set XIP (execute in place)
      tools/mkimage [-D dtc_options] -f fit-image.its fit-image
```

uImage

./mkimage
-A arm
-O linux
-T kernel
-C none
-a **60008040**
-e **60008000**
-d zImage
$output_name

How to jump to kernel

- Use boot command
  - common/cmd_bootm.c
  - command entry
    - int do_bootm (…)
    - boot_os_fn *boot_fn;
- do_bootm_linux(...)
  - arch/arm/lib/bootm.c
- kernel_entry(0, machid, bd->bi_boot_params);
  - **0**
  - **Mach ID**
  - **atag**

Linux Atag

- Kernel parameters

Tag name	Value	Size	Description
ATAG_NONE	0x00000000	2	Empty tag used to end list
ATAG_CORE	0x54410001	5 (2 if empty)	First tag used to start list
ATAG_MEM	0x54410002	4	Describes a physical area of memory
ATAG_VIDEOTEXT	0x54410003	5	Describes a VGA text display
ATAG_RAMDISK	0x54410004	5	Describes how the ramdisk will be used in kernel
ATAG_INITRD2	0x54420005	4	Describes where the compressed ramdisk image is placed in memory
ATAG_SERIAL	0x54410006	4	64 bit board serial number
ATAG_REVISION	0x54410007	3	32 bit board revision number
ATAG_VIDEOLFB	0x54410008	8	Initial values for vesafb-type framebuffers
ATAG_CMDLINE	0x54410009	2 + ((length_of_cmdline + 3) / 4)	Command line to pass to kernel

Let's Exercise

Build u-boot

- Clean
  - **#make distclean**
- modify configure
  - **include/configs/tiny4412.h**
- Do EVB configure
  - **#make tiny4412_defconfig**

Download and boot kernel