arch/arm/lib/board.c
与
common/board_f.c
common/board_r.c
之间的关系

in u-boot/README

```
- CONFIG_SYS_GENERIC_BOARD
This selects the architecture-generic board system instead of the
architecture-specific board files. It is intended to move boards
to this new framework over time. Defining this will disable the
arch/foo/lib/board.c file and use common/board_f.c and
common/board_r.c instead. To use this option your architecture
must support it (i.e. must define __HAVE_ARCH_GENERIC_BOARD in
its config.mk file). If you find problems enabling this option on
your board please report the problem and send patches!
```

即arch/arm/lib/board.c与common/下的board_x.c是互斥的。

如果定义了__HAVE_ARCH_GENERIC_BOARD，则使用common/board_x.c files.

在pegmatite SoC中并没有使用“generic board”这个feature。(在u-boot 2014.1 version)
但在新的u-boot中(u-boot 2015.10 version),已经remove `arch/arm/lib/board.c` .

在"generic board"中已经把初始化分成两个独立的文件
common/board_f.c
common/board_r.c

而在arch/arm/lib/board.c中则是分为两个function.

```
void board_init_f(ulong bootflag);

void board_init_r(gd_t *id, ulong dest_addr);
```

board_init_f()是在u-boot在relocate self以前的board init,而board_init_r()则是在relocate self之后。board_init_f()更早运行。

in arch/arm/lib/crt0.S

```
ENTRY(_main)

/*
* Set up initial C runtime environment and call board_init_f(0).
*/

#if defined(CONFIG_SPL_BUILD) && defined(CONFIG_SPL_STACK)
ldr sp, =(CONFIG_SPL_STACK)
#else
ldr sp, =(CONFIG_SYS_INIT_SP_ADDR)
#endif
bic sp, sp, #7  /* 8-byte alignment for ABI compliance */
sub sp, sp, #GD_SIZE    /* allocate one GD above SP */
bic sp, sp, #7  /* 8-byte alignment for ABI compliance */
mov r9, sp  /* GD is above SP */
mov r0, #0
bl  board_init_f

#if ! defined(CONFIG_SPL_BUILD)

/*
* Set up intermediate environment (new sp and gd) and call
* relocate_code(addr_moni). Trick here is that we'll return
* 'here' but relocated.
*/

ldr sp, [r9, #GD_START_ADDR_SP] /* sp = gd->start_addr_sp */
bic sp, sp, #7  /* 8-byte alignment for ABI compliance */
ldr r9, [r9, #GD_BD]    /* r9 = gd->bd */
sub r9, r9, #GD_SIZE    /* new GD is below bd */

adr lr, here
ldr r0, [r9, #GD_RELOC_OFF] /* r0 = gd->reloc_off */
add lr, lr, r0
ldr r0, [r9, #GD_RELOCADDR] /* r0 = gd->relocaddr */
b   relocate_code
here:

/* Set up final (full) environment */

bl  c_runtime_cpu_setup /* we still call old routine here */

ldr r0, =__bss_start    /* this is auto-relocated! */
ldr r1, =__bss_end  /* this is auto-relocated! */

mov r2, #0x00000000 /* prepare zero to clear BSS */

clbss_l:cmp r0, r1  /* while not at end of BSS */
strlo   r2, [r0]    /* clear 32-bit BSS word */
addlo   r0, r0, #4  /* move to next */
blo clbss_l

bl coloured_LED_init
bl red_led_on

/* call board_init_r(gd_t *id, ulong dest_addr) */
mov     r0, r9                  /* gd_t */
ldr r1, [r9, #GD_RELOCADDR] /* dest_addr */
/* call board_init_r */
ldr pc, =board_init_r   /* this is auto-relocated! */

/* we should not return here. */

#endif

ENDPROC(_main)
```

粗略的看_main()，就分为3 steps

- board_init_f()
- relocate_code() // relocate u-boot self
- board_init_r()