Kiyi

Empowering ARM Embedded Systems

勤研

助您开发ARM嵌入式系统

U－B O O T程序的运行机制

Application Note

AN-日期（如050109）

050718

www.kiyi.com.cn

www.armzone.com

勤研 9200 的 u-boot 的版本号：1.1.1，为目前市场最新的 9200 开发板的 u-boot 的版本号

u-boot 的最新版本代码是 1.1.2，可以到 http://sourceforge.net/projects/u-boot 下载。

u-boot 中已经对 ATMEL 官方的评估板 at91rm9200dk 进行了支持，可参考 u-boot 源代码中的 board\at91rm9200dk 目录下的代码进行修改，以适合你自己的 at91rm9200 开发板。国内通常所谓的 u-boot 移植也就是这样，对 u-boot 完全不支持的一款评估板进行移植，还是有一些工作量的。uboot 主要的移植工作量就在和硬件打交道的那层，就是 board 目录下的文件。首先要读懂参考板的代码， 比较你自己的开发板的外设和 ATMEL 官方的评估板 at91rm9200dk（参考板）有哪些不同，以期望以最小的改动完成移植。

虽然 u-boot 支持很多种评估系统，但是它们的工作流程还是有很多的共通点，这归功于 u-boot 的设计者缜密的结构设计。下面，我以前在工作中做的 ppc 的移植为例，讲一下 u-boot 大致的执行顺序：

1. start.S

System Reset codes
L2 cache init

2. 计算绝对地址 (r3), jump in_flash

board_asm_init: board_asm_init.S

3. setup_bats: start.S

4. 开启 PowerPC 的 MMU (data & instruction)

enable_addr_trans: start.S

5. 开启 L1 data cache, setup stack pointer

l1dcache_enable: cache.S

6. (PowerPC 的 assembly macros 文件中)

GET_GOT: ppc_asm.tmpl

7. PowerPC 前期初始化, 传递 bootflag

cpu_init_f: cpu_init.c

8. board 前期初始化

board_init_f: lib_ppc/board.c
- 提供 (serial) console
- 初始化 RAM
- 重建 stack
- relocate_code: start.S: in_ram

9. board 后期初始化

board_init_r: lib_ppc/board.c

10. 进入 U-Boot command loop

进入 main_loop(): main.c

针对 ARM S3C2410 (对 AT91RM9200 也是一样的，因为它也是 ARM920T 的 Core)，那么 u-boot 程序的入口在 cpu/arm920t 的 start.S，不论针对哪个体系结构，u-boot 总是从 cpu 目录下相应的体系结构目录下的 start.S 这个汇编文件开始的。这个文件中下面这句：

ldr pc, _start_armboot

_start_armboot: .word start_armboot

表示 u-boot 程序已经完成了前面那段汇编(CPU 的初始化)，将跳转到 C 程序中间了。start_armboot 就是一个 C 函数，也是汇编跳转到 C，执行的第一个 C 函数。start_armboot 位于 lib_arm/board.c，完成 ARM 评估板所有的设备初始化，然后进入到 main_loop 函数循环，其实我们要移植的部分就集中在 start_armboot 中所涉及到的各个设备初始化函数，例如 board_init、serial_init、console_init_f、console_init_r 、flash_init ()等等，这些函数和你的评估板相关的。

```
init_fnc_t *init_sequence[] = {
cpu_init, /* basic cpu dependent setup */
board_init, /* basic board dependent setup */
interrupt_init, /* set up exceptions */
env_init, /* initialize environment */
init_baudrate, /* initialze baudrate settings */
serial_init, /* serial communications setup */
console_init_f, /* stage 1 init of console */
display_banner, /* say that we are here */
dram_init, /* configure available RAM banks */
display_dram_config,
#if defined(CONFIG_VCMA9)
checkboard,
#endif
NULL,
};

void start_armboot (void)
{
DECLARE_GLOBAL_DATA_PTR;

ulong size;
init_fnc_t **init_fnc_ptr;
char *s;
#if defined(CONFIG_VFD)
unsigned long addr;
#endif
```

```
/* Pointer is writable since we allocated a register for it */
gd = (gd_t*)(_armboot_start - CFG_MALLOC_LEN - sizeof(gd_t));
memset ((void*)gd, 0, sizeof (gd_t));
gd->bd = (bd_t*)((char*)gd - sizeof(bd_t));
memset (gd->bd, 0, sizeof (bd_t));

monitor_flash_len = _bss_start - _armboot_start;

for (init_fnc_ptr = init_sequence; *init_fnc_ptr; ++init_fnc_ptr) {
if ((*init_fnc_ptr)() != 0) {
hang ();
}
}

/* configure available FLASH banks */
size = flash_init ();
display_flash_config (size);

#ifdef CONFIG_VFD
# ifndef PAGE_SIZE
# define PAGE_SIZE 4096
# endif
/*
* reserve memory for VFD display (always full pages)
*/
/* armboot_end is defined in the board-specific linker script */
addr = (_bss_start + (PAGE_SIZE - 1)) & ~(PAGE_SIZE - 1);
size = vfd_setmem (addr);
gd->fb_base = addr;
#endif /* CONFIG_VFD */

/* armboot_start is defined in the board-specific linker script */
mem_malloc_init (_armboot_start - CFG_MALLOC_LEN);

#if (CONFIG_COMMANDS & CFG_CMD_NAND)
puts ("NAND:");
nand_init(); /* go init the NAND */
#endif

#ifdef CONFIG_HAS_DATAFLASH
AT91F_DataflashInit();
dataflash_print_info();
#endif

/* initialize environment */
```

```
env_relocate ();

#ifdef CONFIG_VFD
/* must do this after the framebuffer is allocated */
drv_vfd_init();
#endif /* CONFIG_VFD */

/* IP Address */
gd->bd->bi_ip_addr = getenv_IPaddr ("ipaddr");

/* MAC Address */
{
int i;
ulong reg;
char *s, *e;
uchar tmp[64];

i = getenv_r ("ethaddr", tmp, sizeof (tmp));
s = (i > 0) ? tmp : NULL;

for (reg = 0; reg < 6; ++reg) {
gd->bd->bi_enetaddr[reg] = s ? simple_strtoul (s, &e, 16) : 0;
if (s)
s = (*e) ? e + 1 : e;
}
}

devices_init (); /* get the devices list going. */

jumptable_init ();

console_init_r (); /* fully init console as a device */

#if defined(CONFIG_MISC_INIT_R)
/* miscellaneous platform dependent initialisations */
misc_init_r ();
#endif

/* enable exceptions */
enable_interrupts ();

/* Perform network card initialisation if necessary */
#ifdef CONFIG_DRIVER_CS8900
cs8900_get_enetaddr (gd->bd->bi_enetaddr);
#endif
```

```
#ifdef CONFIG_DRIVER_LAN91C96
if (getenv ("ethaddr")) {
smc_set_mac_addr(gd->bd->bi_enetaddr);
}
/* eth_hw_init(); */
#endif /* CONFIG_DRIVER_LAN91C96 */

/* Initialize from environment */
if ((s = getenv ("loadaddr")) != NULL) {
load_addr = simple_strtoul (s, NULL, 16);
}
#if (CONFIG_COMMANDS & CFG_CMD_NET)
if ((s = getenv ("bootfile")) != NULL) {
copy_filename (BootFile, s, sizeof (BootFile));
}
#endif /* CFG_CMD_NET */

#ifdef BOARD_LATE_INIT
board_late_init ();
#endif

/* main_loop() can return to retry autoboot, if so just run it again. */
for (;;) {
main_loop ();
}

/* NOTREACHED - no way out of command loop except booting */
}
```

在使用该文档时若遇到什么问题或您有什么意见建议
请致电：021-51097571-805 或 email 至 support@armzone.com