```
/*
 * (C) Copyright 2002
 * Wolfgang Denk, DENX Software Engineering, wd@denx.de.
 *
 * (C) Copyright 2002
 * Sysgo Real-Time Solutions, GmbH <www.elinos.com>
 * Marius Groeger <mgroeger@sysgo.de>
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <common.h>
#include <command.h>
#include <malloc.h>
#include <devices.h>
#include <version.h>
#include <net.h>

#if (CONFIG_COMMANDS & CFG_CMD_NAND)
void nand_init (void);
#endif

ulong monitor_flash_len;

#ifdef CONFIG_HAS_DATAFLASH
extern int  AT91F_DataflashInit(void);
extern void dataflash_print_info(void);
#endif

#ifndef CONFIG_IDENT_STRING
#define CONFIG_IDENT_STRING ""
#endif

const char version_string[] =
    U_BOOT_VERSION" (" __DATE__ " - " __TIME__ ")"CONFIG_IDENT_STRING;

#ifdef CONFIG_DRIVER_CS8900
extern void cs8900_get_enetaddr (uchar * addr);
#endif

#ifdef CONFIG_DRIVER_CS8900
extern void rtl8019_get_enetaddr (uchar * addr);
#endif

#ifdef CONFIG_DRIVER_LAN91C96
#include "../drivers/lan91c96.h"
#endif


/* ghcstop */
/*
 * Begin and End of memory area for malloc(), and current "brk"
 */
static ulong mem_malloc_start = 0;
static ulong mem_malloc_end = 0;
static ulong mem_malloc_brk = 0;

/*
 * cpu/arm920t/start.S
 * _armboot_real_end is the first usable RAM address behind armboot
 * and the various stacks
 _armboot_real_end는 armboot와 여러가지 스택들 뒤에 있는 처음으로 사용가능한 ram address가
 된다. start_armboot에서 호출되는 cpu/arm920t/cpu.c의 cpu_init()함수에서 설정이 된다.
 실제로는  armboot의 코드와 스택크기를 더한값 다음번지이다.
 _armboot_real_end = _armboot_end + CONFIG_STACKSIZE;

```

```
 83  입력으로 들어오는 dest_addr이 _armboot_real_end이다.
 84
 85  malloc영역을 잡는다. 위의 설명을 보면 malloc()으로 사용될 메모리 영역의 시작과
 86  끝을 만들고, 현재의 malloc위치를 brk라고 표시한다....^^
 87  */
 88 static
 89 void mem_malloc_init (ulong dest_addr)
 90 {
 91     mem_malloc_start = dest_addr;
 92     mem_malloc_end = dest_addr + CFG_MALLOC_LEN;
 93     mem_malloc_brk = mem_malloc_start;
 94
 95     memset ((void *) mem_malloc_start, 0,
 96             mem_malloc_end - mem_malloc_start);
 97 }
 98
 99 void *sbrk (ptrdiff_t increment)
100 {
101     ulong old = mem_malloc_brk;
102     ulong new = old + increment;
103
104     if ((new < mem_malloc_start) || (new > mem_malloc_end)) {
105         return (NULL);
106     }
107     mem_malloc_brk = new;
108
109     return ((void *) old);
110 }
111
112 /************************************************************************
113  * Init Utilities                                *
114  ************************************************************************
115  * Some of this code should be moved into the core functions,
116  * or dropped completely,
117  * but let's get it working (again) first...
118  */
119 /* ghcstop */
120 static int init_baudrate (void)
121 {
122     DECLARE_GLOBAL_DATA_PTR;
123
124     uchar tmp[64];  /* long enough for environment variables */
125     /*
126      * board.c, 환경변수인 env_ptr->data혹은 default_environment[]에서
127      * baudrate를 읽어서 gd에 세팅한다.
128      */
129     int i = getenv_r ("baudrate", tmp, sizeof (tmp));
130
131     /*
132      * 읽어온 character(문자...)의 갯수가 0보다 클경우는 읽기에 성공한거고(
133      * 즉, board configuration header file에 CONFIG_BAUDRATE이 선언되어
134      * 있어서 설정이 되어 있는경우) 아니면 직접 CONFIG_BAUDRATE를 board
135      * description structure에 대입을 한다.
136      */
137
138     gd->bd->bi_baudrate = gd->baudrate = (i > 0)
139             ? (int) simple_strtoul (tmp, NULL, 10)
140             : CONFIG_BAUDRATE;
141
142     return (0);
143 }
144
145 static int display_banner (void)
146 {
147     printf ("\n\n%s\n\n", version_string);
148     printf ("U-Boot code: %08lX -> %08lX  BSS: -> %08lX\n",
149         _armboot_start, _bss_start, _bss_end);
150 #ifdef CONFIG_MODEM_SUPPORT
151     puts ("Modem Support enabled\n");
152 #endif
153 #ifdef CONFIG_USE_IRQ
154     printf ("IRQ Stack: %08lx\n", IRQ_STACK_START);
155     printf ("FIQ Stack: %08lx\n", FIQ_STACK_START);
156 #endif
157
158     return (0);
159 }
160
161 /*
162  * WARNING: this code looks "cleaner" than the PowerPC version, but
163  * has the disadvantage that you either get nothing, or everything.
164  * On PowerPC, you might see "DRAM: " before the system hangs - which
```

```
 * gives a simple yet clear indication which part of the
 * initialization if failing.
 */
static int display_dram_config (void)
{
    DECLARE_GLOBAL_DATA_PTR;
    int i;

    puts ("RAM Configuration:\n");

    for(i=0; i<CONFIG_NR_DRAM_BANKS; i++) {
        printf ("Bank #%d: %08lx ", i, gd->bd->bi_dram[i].start);
        print_size (gd->bd->bi_dram[i].size, "\n");
    }

    return (0);
}

static void display_flash_config (ulong size)
{
    puts ("Flash: ");
    print_size (size, "\n");
}


/*
 * Breathe some life into the board...
 *
 * Initialize a serial port as console, and carry out some hardware
 * tests.
 *
 * The first part of initialization is running from Flash memory;
 * its main purpose is to initialize the RAM so that we
 * can relocate the monitor code to RAM.
 */

/*
 * All attempts to come up with a "common" initialization sequence
 * that works for all boards and architectures failed: some of the
 * requirements are just _too_ different. To get rid of the resulting
 * mess of board dependent #ifdef'ed code we now make the whole
 * initialization sequence configurable to the user.
 *
 * The requirements for any new initalization function is simple: it
 * receives a pointer to the "global data" structure as it's only
 * argument, and returns an integer return code, where 0 means
 * "continue" and != 0 means "fatal error, hang the system".
 */
typedef int (init_fnc_t) (void);

/* ghcstop */
init_fnc_t *init_sequence[] = {
    cpu_init,       /* basic cpu dependent setup */   /* => cpu/arm920t/cpu.c */
    board_init,     /* basic board dependent setup */ /* => board/smdk2410/smdk2410.c*/
    interrupt_init,     /* setup internal timer: 이것만*/  /* => cpu/arm920t/interrupts.c */
    /*
     * CFG_ENV_IS_IN_FLASH이 선언되어 있을 경우(smdk2410.h에 선언됨)
     * common/env_flash.c의 env_init()가 호출된다...이 경우도
     * CFG_ENV_ADDR_REDUND이 선언되었느냐 아니냐에 따라 호출되는 함수가 틀려지기는
     * 하나 CFG_ENV_ADDR_REDUND이 smdk2410.h에는 선언이 안되어 있다.
     * CFG_ENV_ADDR_REDUND는 dual env...형태로 쓸 경우에 사용하는 것 같은 느낌이
     * 팍팍든다....
     * ==> common/env_flash.c의 env_init()로 jump...^^
     */
    env_init,       /* initialize environment */
    /*
     * board.c, 환경변수인 env_ptr->data혹은 default_environment[]에서
     * baudrate를 읽어서 gd에 세팅한다.
     */
    init_baudrate,      /* initialze baudrate settings */

    /* ==> cpu/arm920t/serial.c */
    serial_init,        /* serial communications setup */

    /*
     * ==> common/console.c: 그냥 이것은 하는일이 없고 global data
     * structure에 gd->have_console = 1;로 세팅하는 것이다.
     */
    console_init_f,     /* stage 1 init of console */
    /* 잡다한 정보를 뿌려줌...위에 있다... */
    display_banner,     /* say that we are here */

```

```
247     /* ==> board/smdk2410/smdk2410.c */
248     dram_init,      /* configure available RAM banks */
249
250     /* board.c 윗쪽을 볼 것....dram의 뱅크구성에 따라 정보를 뿌려준다. */
251     display_dram_config,
252 #if defined(CONFIG_VCMA9)
253     checkboard,
254 #endif
255     NULL,
256 };
257
258 void start_armboot (void)
259 {
260     /*
261      * include/asm-arm/global_data.h
262      * #define DECLARE_GLOBAL_DATA_PTR     register gd_t *gd asm ("r8")
263      * gd_t형 포인터인 gd를 r8에다 넣는 register변수 지정을 한다(?). 당근 stack변수니깐
264      * 상관은 없겠다...함수 호출해도
265      */
266     DECLARE_GLOBAL_DATA_PTR;
267
268     ulong size;
269     init_fnc_t **init_fnc_ptr; // 함수 배열 선언(즉, 배열을 요기다 가져다가 대입할 생각이군...), 요 바로위에 선
270     char *s;
271 #if defined(CONFIG_VFD)  // lcd controller관련 사항
272     unsigned long addr;
273 #endif
274
275     // 포인터 설정 및 memory clear
276     /* Pointer is writable since we allocated a register for it */
277     gd = (gd_t*)(_armboot_start - CFG_MALLOC_LEN - sizeof(gd_t));
278     memset ((void*)gd, 0, sizeof (gd_t));
279     gd->bd = (bd_t*)((char*)gd - sizeof(bd_t));
280     memset (gd->bd, 0, sizeof (bd_t));
281
282     monitor_flash_len = _bss_start - _armboot_start;
283
284     // 위에 배열로 선언되어 있는 함수들을 일일이 실행시킴...위의 함수 structure를 볼 것.
285     for (init_fnc_ptr = init_sequence; *init_fnc_ptr; ++init_fnc_ptr) {
286         if ((*init_fnc_ptr)() != 0) {
287             hang ();
288         }
289     }
290
291     /* configure available FLASH banks */
292
293     /* ==> board/smdk2410/flash.c: flash 초기화 루틴으로 flash에 따른
294      *  sector갯수와 크기등을 설정하고, 전체 크기를 return해 줌...
295      *  2410에는 amd거와 intel strata flash가 붙어 있고, 점퍼조정을 해서
296      * 부트 플래쉬를 조절 가능하나, 여기서는 amd것을 부트 플래쉬로
297      * 하는 것 같다. 코드에 amd에 대한 것만이 있기 때문이다.
298      */
299     size = flash_init ();
300
301     /* board.c : flash size를 화면에 뿌려줌 */
302     display_flash_config (size);
303
304 /*
305  * LCD관련 define인데 board/trap/vfd.c에 다음과 같이 쓰여있다...아마도 2400용 LCD controller를
306  * 사용할때 쓰는것 같다.
307  * drv_vfd_init(): initialize LCD-Controller of the S3C2400 for using VFDs
308  */
309 #ifdef CONFIG_VFD
310 #   ifndef PAGE_SIZE
311 #     define PAGE_SIZE 4096
312 #   endif
313     /*
314      * reserve memory for VFD display (always full pages)
315      */
316     /* armboot_end is defined in the board-specific linker script */
317     addr = (_bss_start + (PAGE_SIZE - 1)) & ~(PAGE_SIZE - 1);
318     size = vfd_setmem (addr);
319     gd->fb_base = addr;
320 #endif /* CONFIG_VFD */
321         /*
322      * malloc영역을 잡는다. ram에서 armboot코드가 올라간 다음서부터 잡는것 같다.
323      * 위의 코드를 볼 것.
324      */
325     /* armboot_start is defined in the board-specific linker script */
326     mem_malloc_init (_armboot_start - CFG_MALLOC_LEN);
327
328 #if (CONFIG_COMMANDS & CFG_CMD_NAND)
```

```
329     puts ("NAND:");
330     nand_init();        /* go init the NAND */
331 #endif
332
333 #ifdef CONFIG_HAS_DATAFLASH
334     AT91F_DataflashInit();
335     dataflash_print_info();
336 #endif
337
338     /*
339      * initialize environment ==> common/env_common.c
340      * 환경변수의 시작 포인터를 gd->reloc_off이 있을 경우 offset만큼 옮기고
341      * (여기서는 0인것 같으니깐 그대로 일 것 같다)
342      * 만일 env영역이 초기화가 안되었을 경우(valid가 아닐 경우)
343      * default_environment[]를 env영역의 데이터가 들어가는데로 복사해
344      * 넣는다...자세한 것은 필요할 때 보면 될거 같다...여기서 특별히 살펴보지는 않는다.
345      */
346     env_relocate ();
347
348 #ifdef CONFIG_VFD
349     /* must do this after the framebuffer is allocated */
350     drv_vfd_init();
351 #endif /* CONFIG_VFD */
352
353         /* IP Address: string으로된 환경변수(ipaddr=...)를 읽어다가
354     network order byte의 4바이트 값으로 만든다. string이 아닌 hex값으로
355     board description structure에 넣는다.*/
356     gd->bd->bi_ip_addr = getenv_IPaddr ("ipaddr");
357
358     /* MAC Address: 이것도 위의 ipaddr 세팅과 마찬가지이다. */
359     {
360         int i;
361         ulong reg;
362         char *s, *e;
363         uchar tmp[64];
364
365         i = getenv_r ("ethaddr", tmp, sizeof (tmp));
366         s = (i > 0) ? tmp : NULL;
367
368         for (reg = 0; reg < 6; ++reg) {
369             gd->bd->bi_enetaddr[reg] = s ? simple_strtoul (s, &e, 16) : 0;
370             if (s)
371                 s = (*e) ? e + 1 : e;
372         }
373     }
374
375     /* common/devices.c: device linked list를 만들고 serial device등을 등록한다
376     여기서 stdio의 offset을 reloc_off에 따라 조정을 하는 코드가 있는데...
377     reloc_off가 0 같으니깐 거의 그대로인 것 같다...
378     코드가 2중 ptr을 사용하는데...조금은 지금까지 봐왔던 구조와 형태는
379     비슷하지만 사용법은 약간 틀린것 같다.나중에 시간있으면 분석하겠지만
380     지금은 이것의 분석에는 큰 의미를 찾기가 힘들어서 넘어간다...
381
382     그래도 약간의 주석은 달아놨으니 점프해서 볼 것 */
383     devices_init ();    /* get the devices list going. */
384
385     jumptable_init ();
386
387     /* common/console.c: stdout, stdin, stderr를 out,in device로 세팅한다 */
388     console_init_r ();  /* fully init console as a device */
389
390 #if defined(CONFIG_MISC_INIT_R)
391     /* miscellaneous platform dependent initialisations */
392     misc_init_r ();
393 #endif
394
395         /* cpu/arm920t/interrupts.c
396      * enable exceptions
397      * CONFIG_USE_IRQ이 선언 안되어 있을 경우는 그냥 return이다.
398      * smdk2410 보드의 경우 include/configs/smdk2410.h에 #undef로
399      * 되어 있으니깐 그냥 return이다.
400      */
401     /* enable exceptions */
402     enable_interrupts ();
403
404     /*
405     drivers/cs8900.c
406     smdk2410보드에는 이게 달려 있으니깐 이쪽 루틴을 실행한다...
407     당근, smdk2410.h에는 define이 되어 있음.
408     mac address가 환경변수로 저장되어 있을 경우
409     environment에 저장되어 있는 mac address를 읽어서 cs8900의 eeprom에 있는
410     넘이랑 비교해서 다르면 environment에 저장된 값으로 사용하고,
```

```
411     저장되어 있지 않을 경우
412     eeprom에 있는 넘으로 환경변수를 세팅한다.
413     */
414         /* Perform network card initialisation if necessary */
415 #ifdef CONFIG_DRIVER_CS8900
416     cs8900_get_enetaddr (gd->bd->bi_enetaddr);
417 #endif
418
419 #ifdef CONFIG_DRIVER_LAN91C96
420     if (getenv ("ethaddr")) {
421         smc_set_mac_addr(gd->bd->bi_enetaddr);
422     }
423     /* eth_hw_init(); */
424 #endif /* CONFIG_DRIVER_LAN91C96 */
425
426     /* Initialize from environment: default로 image load할 주소, tftp등의
427       명령어를 쓸 경우...근데 난 잘 모르겠다....405할때는 잘 안되더라고 */
428     if ((s = getenv ("loadaddr")) != NULL) {
429         load_addr = simple_strtoul (s, NULL, 16);
430     }
431 #if (CONFIG_COMMANDS & CFG_CMD_NET)
432     if ((s = getenv ("bootfile")) != NULL) {
433         copy_filename (BootFile, s, sizeof (BootFile));
434     }
435 #endif  /* CFG_CMD_NET */
436
437         /* 선언 안되어 있으므로...넘어감 */
438 #ifdef BOARD_LATE_INIT
439     board_late_init ();
440 #endif
441
442     /*
443      * 명령어 입력모드 혹은 autoboot로 들어감....==>common/main.c
444      * : 여기서부터의 분석은 main_loop를 보는게 아니라 개별 명령어들에 대한
445      *   분석을 해야한다.
446      *   다른 명령어는 거의 볼 필요가 없고 부트 명령인 bootm 명령어에 대한
447      *   파일인 common/cmd_bootm.c의 do_bootm()만 보면 된다.
448      * ※ 실제적으로 do_bootm에서 리눅스 커널 이미지 부팅 명령으로 사용되는
449      *    do_bootm_linux()는 ARM일 경우 cmd_bootm.c의 안에 있는 do_bootm_linux()
450      *    코드가 쓰이지 않는다. 거기 있는 것은 ppc용이다.
451      *    ARM용 do_bootm_linux()는 lib_arm/armlinux.c에 있다.
452      */
453     /* main_loop() can return to retry autoboot, if so just run it again. */
454     for (;;) {
455         main_loop ();
456     }
457
458     /* NOTREACHED - no way out of command loop except booting */
459 }
460
461 void hang (void)
462 {
463     puts ("### ERROR ### Please RESET the board ###\n");
464     for (;;);
465 }
466
467 #ifdef CONFIG_MODEM_SUPPORT
468 /* called from main loop (common/main.c) */
469 extern void  dbg(const char *fmt, ...);
470 int mdm_init (void)
471 {
472     char env_str[16];
473     char *init_str;
474     int i;
475     extern char console_buffer[];
476     static inline void mdm_readline(char *buf, int bufsiz);
477     extern void enable_putc(void);
478     extern int hwflow_onoff(int);
479
480     enable_putc(); /* enable serial_putc() */
481
482 #ifdef CONFIG_HWFLOW
483     init_str = getenv("mdm_flow_control");
484     if (init_str && (strcmp(init_str, "rts/cts") == 0))
485         hwflow_onoff (1);
486     else
487         hwflow_onoff(-1);
488 #endif
489
490     for (i = 1;;i++) {
491         sprintf(env_str, "mdm_init%d", i);
492         if ((init_str = getenv(env_str)) != NULL) {
```

```
493             serial_puts(init_str);
494             serial_puts("\n");
495             for(;;) {
496                 mdm_readline(console_buffer, CFG_CBSIZE);
497                 dbg("ini%d: [%s]", i, console_buffer);
498
499                 if ((strcmp(console_buffer, "OK") == 0) ||
500                     (strcmp(console_buffer, "ERROR") == 0)) {
501                     dbg("ini%d: cmd done", i);
502                     break;
503                 } else /* in case we are originating call ... */
504                     if (strncmp(console_buffer, "CONNECT", 7) == 0) {
505                         dbg("ini%d: connect", i);
506                         return 0;
507                     }
508             }
509         } else
510             break; /* no init string - stop modem init */
511
512         udelay(100000);
513     }
514
515     udelay(100000);
516
517     /* final stage - wait for connect */
518     for(;i > 1;) { /* if 'i' > 1 - wait for connection
519                       message from modem */
520         mdm_readline(console_buffer, CFG_CBSIZE);
521         dbg("ini_f: [%s]", console_buffer);
522         if (strncmp(console_buffer, "CONNECT", 7) == 0) {
523             dbg("ini_f: connected");
524             return 0;
525         }
526     }
527
528     return 0;
529 }
530
531 /* 'inline' - We have to do it fast */
532 static inline void mdm_readline(char *buf, int bufsiz)
533 {
534     char c;
535     char *p;
536     int n;
537
538     n = 0;
539     p = buf;
540     for(;;) {
541         c = serial_getc();
542
543         /*      dbg("(%c)", c); */
544
545         switch(c) {
546         case '\r':
547             break;
548         case '\n':
549             *p = '\0';
550             return;
551
552         default:
553             if(n++ > bufsiz) {
554                 *p = '\0';
555                 return; /* sanity check */
556             }
557             *p = c;
558             p++;
559             break;
560         }
561     }
562 }
563 #endif  /* CONFIG_MODEM_SUPPORT */
564
```