```c
/*
 * (C) Copyright 2000
 * Paolo Scaffardi, AIRVENT SAM s.p.a - RIMINI(ITALY), arsenio@tin.it
 *
 * 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 <config.h>
#include <common.h>
#include <stdarg.h>
#include <malloc.h>
#include <devices.h>
#ifdef CONFIG_LOGBUFFER
#include <logbuff.h>
#endif
#if defined(CONFIG_HARD_I2C) || defined(CONFIG_SOFT_I2C)
#include <i2c.h>
#endif

list_t devlist = 0;
device_t *stdio_devices[] = { NULL, NULL, NULL };
char *stdio_names[MAX_FILES] = { "stdin", "stdout", "stderr" };

#if defined(CONFIG_SPLASH_SCREEN) && !defined(CFG_DEVICE_NULLDEV)
#define CFG_DEVICE_NULLDEV  1
#endif


#ifdef CFG_DEVICE_NULLDEV
void nulldev_putc(const char c)
{
  /* nulldev is empty! */
}

void nulldev_puts(const char *s)
{
  /* nulldev is empty! */
}

int nulldev_input(void)
{
  /* nulldev is empty! */
  return 0;
}
#endif

/*****************************************************************************
 * SYSTEM DRIVERS
 *****************************************************************************
 */

static void drv_system_init (void)
{
    device_t dev;

    memset (&dev, 0, sizeof (dev));

    strcpy (dev.name, "serial");
    dev.flags = DEV_FLAGS_OUTPUT | DEV_FLAGS_INPUT | DEV_FLAGS_SYSTEM;
#if CONFIG_SERIAL_SOFTWARE_FIFO
    dev.putc = serial_buffered_putc;
    dev.puts = serial_buffered_puts;
    dev.getc = serial_buffered_getc;
    dev.tstc = serial_buffered_tstc;
#else
    dev.putc = serial_putc;
```

```
    dev.puts = serial_puts;
    dev.getc = serial_getc;
    dev.tstc = serial_tstc;
#endif

    device_register (&dev);

#ifdef CFG_DEVICE_NULLDEV
    memset (&dev, 0, sizeof (dev));

    strcpy (dev.name, "nulldev");
    dev.flags = DEV_FLAGS_OUTPUT | DEV_FLAGS_INPUT | DEV_FLAGS_SYSTEM;
    dev.putc = nulldev_putc;
    dev.puts = nulldev_puts;
    dev.getc = nulldev_input;
    dev.tstc = nulldev_input;

    device_register (&dev);
#endif
}

/*****************************************************************************
 * DEVICES
 *****************************************************************************
 */

int device_register (device_t * dev)
{
    ListInsertItem (devlist, dev, LIST_END);
    return 0;
}

/* deregister the device "devname".
 * returns 0 if success, -1 if device is assigned and 1 if devname not found
 */
#ifdef  CFG_DEVICE_DEREGISTER
int device_deregister(char *devname)
{
    int i,l,dev_index;
    device_t *dev = NULL;
    char temp_names[3][8];

    dev_index=-1;
    for (i=1; i<=ListNumItems(devlist); i++) {
        dev = ListGetPtrToItem (devlist, i);
        if(strcmp(dev->name,devname)==0) {
            dev_index=i;
            break;
        }
    }
    if(dev_index<0) /* device not found */
        return 0;
    /* get stdio devices (ListRemoveItem changes the dev list) */
    for (l=0 ; l< MAX_FILES; l++) {
        if (stdio_devices[l] == dev) {
            /* Device is assigned -> report error */
            return -1;
        }
        memcpy (&temp_names[l][0],
            stdio_devices[l]->name,
            sizeof(stdio_devices[l]->name));
    }
    ListRemoveItem(devlist,NULL,dev_index);
    /* reassign Device list */
    for (i=1; i<=ListNumItems(devlist); i++) {
        dev = ListGetPtrToItem (devlist, i);
        for (l=0 ; l< MAX_FILES; l++) {
            if(strcmp(dev->name,temp_names[l])==0) {
                stdio_devices[l] = dev;
            }
        }
    }
    return 0;
}
#endif  /* CFG_DEVICE_DEREGISTER */

/* ghcstop */
/* device에 대한 링크드 리스트를 작성한다. 시리얼 디바이스를 등록한다 */
int devices_init (void)
{
    DECLARE_GLOBAL_DATA_PTR;

```

```
165 /* CONFIG_ARM이 선언 안되어 있으므로 코드는 동작한다...*/
166 #ifndef CONFIG_ARM     /* already relocated for current ARM implementation */
167     ulong relocation_offset = gd->reloc_off;
168     int i;
169
170     /* relocate device name pointers */
171     for (i = 0; i < (sizeof (stdio_names) / sizeof (char *)); ++i) {
172         stdio_names[i] = (char *) (((ulong) stdio_names[i]) +
173                         relocation_offset);
174     }
175 #endif
176
177     /* Initialize the list */
178     devlist = ListCreate (sizeof (device_t));
179
180     if (devlist == NULL) {
181         eputs ("Cannot initialize the list of devices!\n");
182         return -1;
183     }
184 #if defined(CONFIG_HARD_I2C) || defined(CONFIG_SOFT_I2C)
185     i2c_init (CFG_I2C_SPEED, CFG_I2C_SLAVE);
186 #endif
187 #ifdef CONFIG_LCD
188     drv_lcd_init ();
189 #endif
190 #if defined(CONFIG_VIDEO) || defined(CONFIG_CFB_CONSOLE)
191     drv_video_init ();
192 #endif
193 #ifdef CONFIG_KEYBOARD
194     drv_keyboard_init ();
195 #endif
196 #ifdef CONFIG_LOGBUFFER
197     drv_logbuff_init ();
198 #endif
199     drv_system_init ();/* 여기서 serial device에 대한 코드가 등록된다.
200                           serial putc, puts...등등...., 위의 코드를 볼 것*/
201
202     /* device가 초기화 되었다는 flag를 넣는다...*/
203     gd-> flags |= GD_FLG_DEVINIT;   /* device initialization done */
204
205     return (0);
206 }
207
208 int devices_done (void)
209 {
210     ListDispose (devlist);
211
212     return 0;
213 }
214
```