```
/*
 *    Filename: cfag12864b-example.c
 *     Version: 0.1.0
 * Description: cfag12864b LCD userspace example program
 *     License: GPLv2
 *
 *      Author: Copyright (C) Miguel Ojeda Sandonis
 *        Date: 2006-10-31
 *
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License version 2 as
 *  published by the Free Software Foundation.
 *
 *  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
 *
 */

/*
 * ------------------------
 * start of cfag12864b code
 * ------------------------
 */

#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>

#define CFAG12864B_WIDTH                (128)
#define CFAG12864B_HEIGHT               (64)
#define CFAG12864B_SIZE                 (128 * 64 / 8)
#define CFAG12864B_BPB                  (8)
#define CFAG12864B_ADDRESS(x, y)        ((y) * CFAG12864B_WIDTH / \
                                        CFAG12864B_BPB + (x) / CFAG12864B_BPB)
#define CFAG12864B_BIT(n)               (((unsigned char) 1) << (n))

#undef CFAG12864B_DOCHECK
#ifdef CFAG12864B_DOCHECK
        #define CFAG12864B_CHECK(x, y)          ((x) < CFAG12864B_WIDTH && \
                                                (y) < CFAG12864B_HEIGHT)
#else
        #define CFAG12864B_CHECK(x, y)          (1)
#endif

int cfag12864b_fd;
unsigned char * cfag12864b_mem;
unsigned char cfag12864b_buffer[CFAG12864B_SIZE];
```

```
/*
 * init a cfag12864b framebuffer device
 *
 * No error:       return = 0
 * Unable to open: return = -1
 * Unable to mmap: return = -2
 */
static int cfag12864b_init(char *path)
{
        cfag12864b_fd = open(path, O_RDWR);
        if (cfag12864b_fd == -1)
                return -1;

        cfag12864b_mem = mmap(0, CFAG12864B_SIZE, PROT_READ | PROT_WRITE,
                MAP_SHARED, cfag12864b_fd, 0);
        if (cfag12864b_mem == MAP_FAILED) {
                close(cfag12864b_fd);
                return -2;
        }

        return 0;
}

/*
 * exit a cfag12864b framebuffer device
 */
static void cfag12864b_exit(void)
{
        munmap(cfag12864b_mem, CFAG12864B_SIZE);
        close(cfag12864b_fd);
}

/*
 * set (x, y) pixel
 */
static void cfag12864b_set(unsigned char x, unsigned char y)
{
        if (CFAG12864B_CHECK(x, y))
                cfag12864b_buffer[CFAG12864B_ADDRESS(x, y)] |=
                        CFAG12864B_BIT(x % CFAG12864B_BPB);
}

/*
 * unset (x, y) pixel
 */
static void cfag12864b_unset(unsigned char x, unsigned char y)
{
        if (CFAG12864B_CHECK(x, y))
                cfag12864b_buffer[CFAG12864B_ADDRESS(x, y)] &=
                        ~CFAG12864B_BIT(x % CFAG12864B_BPB);
}

/*
 * is set (x, y) pixel?
 *
 * Pixel off: return = 0
 * Pixel on:  return = 1
 */
static unsigned char cfag12864b_isset(unsigned char x, unsigned char y)
{
	if (CFAG12864B_CHECK(x, y))
		if (cfag12864b_buffer[CFAG12864B_ADDRESS(x, y)] &
			CFAG12864B_BIT(x % CFAG12864B_BPB))
			return 1;

	return 0;
}

/*
 * not (x, y) pixel
 */
static void cfag12864b_not(unsigned char x, unsigned char y)
{
	if (cfag12864b_isset(x, y))
		cfag12864b_unset(x, y);
	else
		cfag12864b_set(x, y);
}

/*
 * fill (set all pixels)
 */
static void cfag12864b_fill(void)
{
	unsigned short i;

	for (i = 0; i < CFAG12864B_SIZE; i++)
		cfag12864b_buffer[i] = 0xFF;
}

/*
 * clear (unset all pixels)
 */
static void cfag12864b_clear(void)
{
	unsigned short i;

	for (i = 0; i < CFAG12864B_SIZE; i++)
		cfag12864b_buffer[i] = 0;
}

/*
 * format a [128*64] matrix
 *
 * Pixel off: src[i] = 0
 * Pixel on:  src[i] > 0
 */
static void cfag12864b_format(unsigned char * matrix)
{
	unsigned char i, j, n;
```

```
        for (i = 0; i < CFAG12864B_HEIGHT; i++)
        for (j = 0; j < CFAG12864B_WIDTH / CFAG12864B_BPB; j++) {
                cfag12864b_buffer[i * CFAG12864B_WIDTH / CFAG12864B_BPB +
                        j] = 0;
                for (n = 0; n < CFAG12864B_BPB; n++)
                        if (matrix[i * CFAG12864B_WIDTH +
                                j * CFAG12864B_BPB + n])
                                cfag12864b_buffer[i * CFAG12864B_WIDTH /
                                        CFAG12864B_BPB + j] |=
                                        CFAG12864B_BIT(n);
        }
}

/*
 * blit buffer to lcd
 */
static void cfag12864b_blit(void)
{
        memcpy(cfag12864b_mem, cfag12864b_buffer, CFAG12864B_SIZE);
}

/*
 * ----------------------
 * end of cfag12864b code
 * ----------------------
 */

#include <stdio.h>

#define EXAMPLES        6

static void example(unsigned char n)
{
        unsigned short i, j;
        unsigned char matrix[CFAG12864B_WIDTH * CFAG12864B_HEIGHT];

        if (n > EXAMPLES)
                return;

        printf("Example %i/%i - ", n, EXAMPLES);

        switch (n) {
        case 1:
                printf("Draw points setting bits");
                cfag12864b_clear();
                for (i = 0; i < CFAG12864B_WIDTH; i += 2)
                        for (j = 0; j < CFAG12864B_HEIGHT; j += 2)
                                cfag12864b_set(i, j);
                break;

        case 2:
                printf("Clear the LCD");
                cfag12864b_clear();
                break;

        case 3:
```

```
		printf("Draw rows formatting a [128*64] matrix");
		memset(matrix, 0, CFAG12864B_WIDTH * CFAG12864B_HEIGHT);
		for (i = 0; i < CFAG12864B_WIDTH; i++)
			for (j = 0; j < CFAG12864B_HEIGHT; j += 2)
				matrix[j * CFAG12864B_WIDTH + i] = 1;
		cfag12864b_format(matrix);
		break;

	case 4:
		printf("Fill the lcd");
		cfag12864b_fill();
		break;

	case 5:
		printf("Draw columns unsetting bits");
		for (i = 0; i < CFAG12864B_WIDTH; i += 2)
			for (j = 0; j < CFAG12864B_HEIGHT; j++)
				cfag12864b_unset(i, j);
		break;

	case 6:
		printf("Do negative not-ing all bits");
		for (i = 0; i < CFAG12864B_WIDTH; i++)
			for (j = 0; j < CFAG12864B_HEIGHT; j ++)
				cfag12864b_not(i, j);
		break;
	}

	puts(" - [Press Enter]");
}

int main(int argc, char *argv[])
{
	unsigned char n;

	if (argc != 2) {
		printf(
			"Sintax:  %s fbdev\n"
			"Usually: /dev/fb0, /dev/fb1...\n", argv[0]);
		return -1;
	}

	if (cfag12864b_init(argv[1])) {
		printf("Can't init %s fbdev\n", argv[1]);
		return -2;
	}

	for (n = 1; n <= EXAMPLES; n++) {
		example(n);
		cfag12864b_blit();
		while (getchar() != '\n');
	}

	cfag12864b_exit();

	return 0;
```

```
}
```