Lab 6-1: I/O Control for PXA270

Yu-Lun Huang

2020-08-13

There is an 8-bit LED lamps on the motherboard of PXA270, numbered frome D9(1) to D16(8) We use the `creator-pxa270-lcd.ko` module to control the LED lamps, it is also used to drive the LCD, 7-segement LED, KeyPAD, and DIP Switch.

1 Compile Modules

Rebuild your kernel and rootfile system to support `creator-pxa270-lcd.ko` module:

- Get the source code of `creator-pxa270-lcd.ko`:
  Download `Creator_PXA270_LCD_Device_Driver.src.tar.gz` from E3 website and decompressed it to your kernel source.

  ```
  SHELL> cd ~
  SHELL> tar xzvf Creator_PXA270_LCD_Device_Driver.src.tar.gz
  ```

- Configure kernel source:

  ```
  SHELL> cd ~/microtime/linux
  SHELL> sed -i'452s/.*/%&/' Makefile
  SHELL> make mrproper
  SHELL> make menuconfig
  ```

  – In the window of “Linux Kernel Configuration”, select “Load an Alternate Configuration from File” and load the configuration file `arch/arm/configs/creator_pxa270_defconfig`.
  – Select “Device Drivers” → “Character devices” and mark “Creator-pxa270 LCD” as `[M]`.
  – Save and exit kernel configuration.

- Make Image:
  Compile Linux kernel and `creator-pxa270-lcd.ko` module.

  ```
  SHELL> make clean
  SHELL> make
  ```

  The `creator-pxa270-lcd.ko` module will be placed at `microtime/linux/drivers/char/`.

- Make new root filesystem:
  Copy `creator-pxa270-lcd.ko` module into root filesystem.

```
SHELL> cp ~/microtime/linux/drivers/char/creator-pxa270-lcd.ko
~/microtime/rootfs/lib/modules/2.6.15.3/kernel/drivers/char/
```

Then, rebuild and flash root filesystem.

2 Load Modules

Type the following command to load the `creator-pxa270-lcd.ko` on PXA270.

```
> insmod lib/modules/2.6.15.3/kernel/drivers/char/creator-pxa270-lcd.ko
```

3 PXA I/O: Control LED

- LED programming guide

  Header file:

```c
#include "asm-arm/arch-pxa/lib/creator_pxa270_lcd.h"
```

  Commands:

```
LED_IOCTL_SET          // set the specified LED (D9 - D16)
LED_IOCTL_CLEAR        // clear the specified LED (D9 - D16)
```

  Values:

```
LED_ALL_ON          0xFF
LED_ALL_OFF         0x00
LED_D9_INDEX        1
LED_D10_INDEX       2
LED_D11_INDEX       3
LED_D12_INDEX       4
LED_D13_INDEX       5
LED_D14_INDEX       6
LED_D15_INDEX       7
LED_D16_INDEX       8
```

  Sample code:

```c
/*
 * led.c -- the sample code for controlling LEDs on Creator.
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include "asm-arm/arch-pxa/lib/creator_pxa270_lcd.h"

int main(int argc, char *argv[])
{
    int fd;         /* file descriptor for /dev/lcd */
    int retval;

    unsigned short data;

    /* Open device /dev/lcd */
    if((fd = open("/dev/lcd", O_RDWR)) < 0)
    {
        printf("Open /dev/lcd faild.\n");
        exit(-1);
    }

    /* Turn on all LED lamps */
    data = LED_ALL_ON;
    ioctl(fd, LED_IOCTL_SET, &data);
    printf("Turn on all LED lamps\n");
    sleep(3);

    /* Turn off all LED lamps */
    data = LED_ALL_OFF;
    ioctl(fd, LED_IOCTL_SET, &data);
    printf("Turn off all LED lamps\n");
    sleep(3);

    /* Turn on D9 */
    data = LED_D9_INDEX;
    ioctl(fd, LED_IOCTL_BIT_SET, &data);
    printf("Turn on D9 \n");
    sleep(3);

    /* Turn off D9 */
    data = LED_D9_INDEX;
    ioctl(fd, LED_IOCTL_BIT_CLEAR, &data);
    printf("Turn off D9 \n");
    sleep(3);

    /* Close fd */
    close(fd);

    return 0;
}
```

Add the header search path when compile `led.c`.

```
SHELL> arm-unknown-linux-gnu-gcc -o led led.c
-L /opt/arm-unknown-linux-gnu/arm-unknown-linux-gnu/lib/
-I /opt/arm-unknown-linux-gnu/arm-unknown-linux-gnu/include/
-I /home/lab616/microtime/linux/include/
```

4 PXA I/O: 7-Segment

- Header files: asm-arm/arch-pxa/lib/creator_pxa270_lcd.h
- Function: ioctl(fd, command, data)
  - Command:
    `_7SEG_IOCTL_ON`: turn on 7 segment LED (no data is needed)
    `_7SEG_IOCTL_OFF`: turn off 7 segment LED (no data is needed)
    `_7SEG_IOCTL_SET`: set 7 segment LED (`_7seg_info_t`)
  - Data:

```c
typedef struct _7Seg_Info{
   unsigned char Mode;  // _7SEG_MODE_PATTERN or _7SEG_MODE_HEX_VALUE
   unsigned char Which; // D5 ~ D8
   unsigned long Value; // pattern or hex
} _7seg_info_t;
```

- The setting of _7Seg_Info:
  - flags used in Mode:

```c
#define _7SEG_MODE_PATTERN        0
#define _7SEG_MODE_HEX_VALUE      1
```

  - flags used in Which:

```c
#define _7SEG_D5_INDEX  8  // Segment D5 (1)
#define _7SEG_D6_INDEX  4  // Segment D6 (2)
#define _7SEG_D7_INDEX  2  // Segment D7 (3)
#define _7SEG_D8_INDEX  1  // Segment D8 (4)
#define _7SEG_ALL
        (_7SEG_D5_INDEX|_7SEG_D6_INDEX|_7SEG_D7_INDEX|_7SEG_D8_INDEX)
```

The following is the sample code of 7 segment display control.

```c
_7seg_info_t    data;
int             fd, ret, i;

if ((fd = open("/dev/lcd", O_RDWR)) < 0)  return (-1);

ioctl(fd, _7SEG_IOCTL_ON, NULL);
data.Mode = _7SEG_MODE_HEX_VALUE;
data.Which = _7SEG_ALL;
data.Value = 0x2004;
ioctl(fd, _7SEG_IOCTL_SET, &data);
sleep (3);
data.Mode = _7SEG_MODE_PATTERN;
data.Which = _7SEG_D5_INDEX | _7SEG_D8_INDEX;
data.Value = 0x6d7f; /* change to 5008 */
ioctl(fd, _7SEG_IOCTL_SET, &data);
ioctl(fd, _7SEG_IOCTL_OFF, NULL);
close(fd);
```

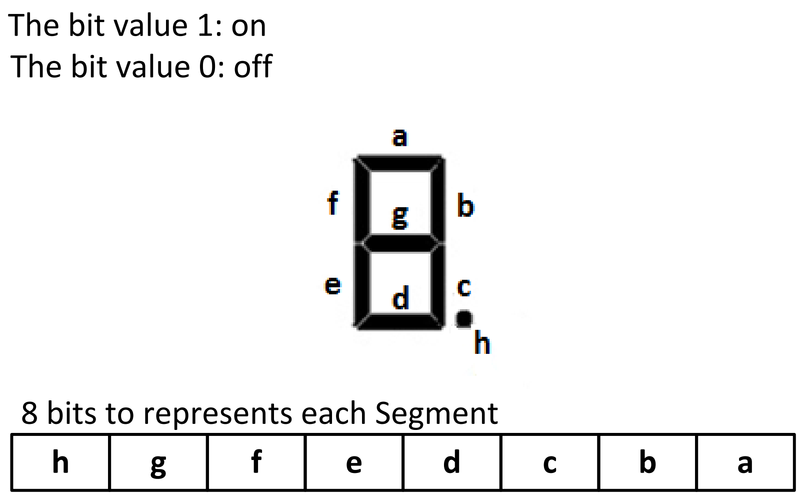

Figure 1: The layout of 7 segment display

5 PXA I/O: Keypad

- Keypad I/O
  - Header files: asm-arm/arch-pxa/lib/creator_pxa270_lcd.h
  - Function: ioctl(fd, command, data)
    - Command:
      `KEY_IOCTL_GET_CHAR`: unsigned short, get its ASCII value.
      `KEY_IOCTL_WAIT_CHAR`: wait until get a character.
      `KEY_IOCTL_CHECK_EMPTY`
      `KEY_IOCTL_CLEAR`
      `KEY_IOCTL_CANCEL_WAIT_CHAR`
    - Definition

```
#define VK_S2  1      /* ASCII = `1' */
#define VK_S3  2      /* ASCII = `2' */
#define VK_S4  3      /* ASCII = `3' */
#define VK_S5  10     /* ASCII = `A' */
#define VK_S6  4
#define VK_S7  5
#define VK_S8  6
#define VK_S9  11
#define VK_S10 7
#define VK_S11 8
#define VK_S12 9
#define VK_S13 12
#define VK_S14 14     /* ASCII = `*' */
#define VK_S15 0
#define VK_S16 15     /* ASCII = `#' */
#define VK_S17 13
```

∗ Sample

```
unsigned short     key;
int           fd, ret;

if ((fd = open("/dev/lcd", O_RDWR)) < 0)  return (-1);

ioctl(fd, KEY_IOCTL_CLEAR, key);
while(1) {
  ret = ioctl(fd, KEY_IOCTL_CHECK_EMTPY, &key)
  if (ret < 0) {
    sleep(1);
    continue;
  }
  ret = ioctl(fd, KEY_IOCTL_GET_CHAR, &key)
  if (key & 0xff) == '#') break;   /* terminate */
}
close(fd);
```

6 PXA I/O: LCD Control

- Header files: asm-arm/arch-pxa/lib/creator__pxa270__lcd.h
- Function: ioctl(fd, command, data)
  - Device Name: `/dev/lcd`
  - Data Structure

```
/* Data structure for writing char to LCD screen */
typedef struct lcd_write_info {
   unsigned char Msg[512];        /* the array for saving input */
   unsigned short Count;          /* the number of input char */
   int CursorX, CursorY;          /* X, Y axis of cursor */
} lcd_write_info_t;

/* Data structure for writing a picture to LCD screen */
typedef struct lcd_full_image_info {
   unsigned short data[0x800];    /* the array for saving picture */
} lcd_full_image_info_t;
```

  - Command

```
/* Clear LCD data and move cursor back to the Upper-left corner */
#define LCD_IOCTL_CLEAR     LCD_IO ( 0x0 )

/* Write char to LCD */
#define LCD_IOCTL_WRITE     LCD_IOW( 0x01, lcd_write_info_t )

/* Turn On or Off cursor */
#define LCD_IOCTL_CUR_ON    LCD_IO( 0x02 )
#define LCD_IOCTL_CUR_OFF   LCD_IO( 0x03 )

```

```
/* Get and Set the position (X, Y) of cursor */
#define LCD_IOCTL_CUR_GET  LCD_IOR( 0x04, lcd_write_info_t )
#define LCD_IOCTL_CUR_SET  LCD_IOW( 0x05, lcd_write_info_t )

/* Write a picture to LCD */
#define LCD_IOCTL_DRAW_FULL_IMAGE LCD_IOW(0x06, lcd_full_image_info_t)
```

The following is the sample code of LCD control.

```
/*
 * lcd.c -- The sample code to print "Hello World" on LCD screen.
 */

#include <stdio.h>
#include <sys/fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include "asm-arm/arch-pxa/lib/creator_pxa270_lcd.h"

int main()
{
    int fd;
    lcd_write_info_t display;  /* struct for saving LCD data */

    /* Open device /dev/lcd */
    if ((fd = open("/dev/lcd",O_RDWR) < 0))
    {
        printf("open /dev/lcd error\n");
        return (-1);
    }

    /* Clear LCD */
    ioctl(fd,LCD_IOCTL_CLEAR,NULL);

    /* Save output string to display data structure */
    display.Count = sprintf((char *) display.Msg, "Hello World\n");
    /* Print out "Hello World" to LCD */
    ioctl(fd, LCD_IOCTL_WRITE, &display);


    /* Get the cursor position */
    ioctl(fd, LCD_IOCTL_CUR_GET, &display);
    printf("The cursor position is at (x,y) = (%d,%d)\n",
          display.CursorX, display.CursorY);

    close(fd);
    return 0;
}
```

7 PXA I/O: Audio Control

- Header files: asm-arm/arch-pxa/lib/creator_pxa270_lcd.h
- Function: ioctl(fd, command);
  - Device Name: `/dev/lcd`
  - Command

```
IOCTL_RECORD_START
IOCTL_RECORD_STOP
IOCTL_PLAY_START
IOCTL_PLAY_STOP
```

The following is the sample code of Audio control.

```
#include <sys/ioctl.h>
#include <stdio.h>
#include <stdlib.h>
#include "creator_s3c4510_codec.h"//must write the absolutely path

#define RECORDING_SIZE 8*8000  /* no of bytes for 4 seconds at 8000 per second */
#define SIZE_16 4*8000   /* no of int for 4 seconds */
main() {
int codec_fd;
int TotalReadSize, nRead, count, i;
char AudioBuffer[RECORDING_SIZE];
FILE *audio_fd;

codec_fd = open( "/dev/codec" , O_RDWR);  /* Open the codec device driver */
if  (codec_fd < 0) {
    printf ( "Open␣/dev/codec␣␣error\n"   );
    return (-1);
}

if( ioctl (codec_fd, IOCTL_RECORD_START) < 0) {  /*Start recording */
    printf ( "Audio␣recording␣start␣error\n"   );
    close(codec_fd);
    return (-1);
}
printf("Say␣something␣to␣the␣microphone\n");
sleep(4);  /* record 4 seconds of data */

if( ioctl (codec_fd, IOCTL_RECORD_STOP) < 0) {  /* Stop recording */
    printf ("Audio␣recording␣stop␣error\n");
    close(codec_fd);
    return (-1);
}
printf("Recording␣stopped\n");

sleep(3);  /* sleep for 3 seconds */

if( ioctl (codec_fd, IOCTL_PLAY_START) < 0) {  /*Start playback */
```

```
    printf ( "Audio␣playback␣start␣error\n");
    close(codec_fd);
    return (-1);
}
printf("Start␣playing␣recorded␣data␣repeatedly\n");
sleep(12);  /* Sleep for 12 seconds to allow repeating twice */

if( ioctl (codec_fd, IOCTL_PLAY_STOP) < 0) {  /*Stop playback */
    printf ( "Audio␣playback␣stop␣error\n");
    close(codec_fd);
    return (-1);
}
printf("Playback␣stopped\n");

/* Begin reading the data*/
TotalReadSize = 0;
count =RECORDING_SIZE;
do {
   if (count + TotalReadSize > RECORDING_SIZE)
        count = RECORDING_SIZE- TotalReadSize ;

   nRead = read(codec_fd, AudioBuffer+TotalReadSize, count);
   if (nRead > 0 )
TotalReadSize += nRead;
   else if (nRead == 0)  /* EOF */
        break;
  else {
        printf("Reading␣audio␣data␣failed!\n");
        close(codec_fd);
        exit(1);
        }
} while (TotalReadSize < RECORDING_SIZE);

/* write the audio data to a file */
audio_fd = fopen("myaudio.txt", "w");
for( i = 0; i < SIZE_16; i = i+2) {
fprintf(audio_fd,"%d\n", (int) AudioBuffer[i]);  /* convert into 16-bit 2's complement */
}

fclose(audio_fd);
close(codec_fd);
}
```