HCRTOS xpt2046 触摸屏使用说明文档

1. 目录

2. 文档履历

版本号	日期	制/修订人	制/修订记录
1.0	2023.4.10	邱浩佳	新增文档说明

3. 概述

3.1 编写目的

介绍xpt2046驱动模块的使用

3.2 读者对象

软件开发工程师和技术支持工程师。

4. 模块介绍

- xpt2046触摸屏为电阻触摸屏，
- 使用spi协议进行数据通信。
- 驱动对接了Linux的input子系统，应用层获取数据通过对应的input节点，就可以获取到数据。

4.1 设备树配置

```
spi-gpio {

        // pinmux-active = <PINPAD_L28 0 PINPAD_L24 0 PINPAD_L23 0 PINPAD_L25 0>;
        status = "okay";

        gpio-sck  = <PINPAD_L28>;
        gpio-mosi = <PINPAD_L24>;

        gpio-miso = <PINPAD_L23>;

        cs-gpios  = <PINPAD_L25>;

        num-chipselects = <1>;



        xpt2046{

                status = "okay";

                touch-gpio = <PINPAD_L27>;

                devpath = "/dev/xpt2046";

        };

};
```

4.2 menuconfig配置

在SDK根目录输入make menuconfig，按照下面的路径进行勾选hy46xx_ts驱动。

```
There is no help available for this option.
Symbol: CONFIG_HC_XPT2046 [=n]
Type  : bool
Prompt: xpt2046
  Location:
    -> Components
      -> kernel (BR2_PACKAGE_KERNEL [=y])
        -> Drivers
          -> input event (CONFIG_DRV_INPUT [=y])
            -> tp menu (CONFIG_TP [=y])
  Defined at tp:1
  Depends on: BR2_PACKAGE_KERNEL [=y] && CONFIG_DRV_INPUT [=y] &&
CONFIG_TP [=y]
```

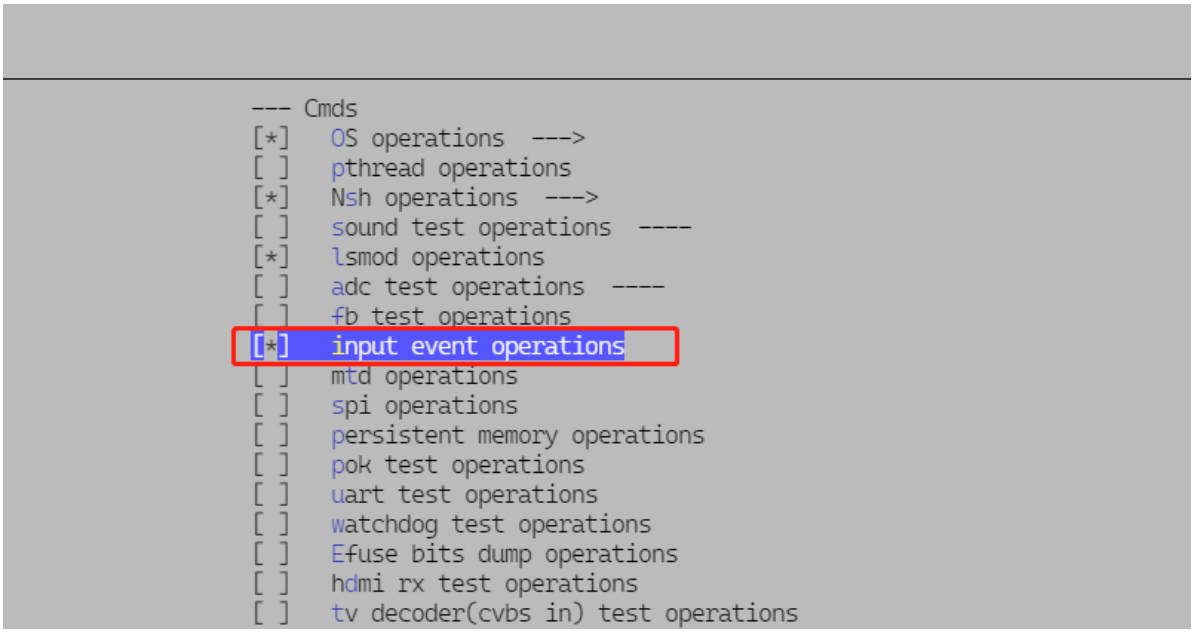
4.3 编译

配置完成后，输入make kernel-rebuild all，进行编译和烧录，在串口控制台输入ls命令就可以看到hy46xx驱动。

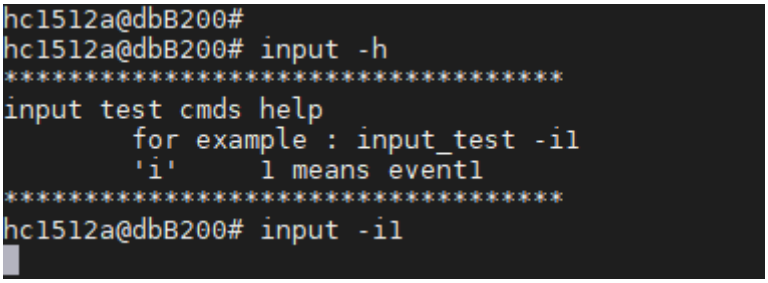
5. 模块测试用例与Sample Code

在SDK根目录输入make menuconfig，根据下面路径选中测试命令

```
Location:
  -> Components
    -> Cmds (BR2_PACKAGE_CMDS [=y])
```



选中后，输入make cmds-rebuild all，进行编译和烧录。在串口控制台输入下面命令即可以查看测试命令。



按压触摸屏就会接收到触摸点的消息。

5.1 测试代码

```c
#include <stdlib.h>

#include <poll.h>
#include <unistd.h>
#include <stddef.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <hcuapi/input.h>
#include <kernel/lib/console.h>

#define BUF_SIZE 1024

static void print_help(void) {
        printf("**********************************\n");
        printf("input test cmds help\n");
        printf("\tfor example : input_test -i1\n");
        printf("\t'i'   1 means event1\n");
        printf("**********************************\n");
}

static int input_test(int argc, char *argv[])
{
        int fd;
        struct input_event t;
        struct pollfd pfd;
        char input_buf[BUF_SIZE];
        char *s = "/dev/input/event";

        long tmp;
        int x = 0, y = 0;
        int event_num = -1;
        char ch;
        opterr = 0;
        optind = 0;

        while((ch = getopt(argc, argv, "hi:")) != EOF){
                switch (ch) {
                        case 'h':
                                print_help();
                                return 0;
                        case 'i':
                                tmp = strtoll(optarg, NULL,10);
```

```
                                event_num = tmp;
                                break;
                        default:
                                printf("Invalid parameter %c\r\n", ch);
                                print_help();
                                return -1;
                }
        }
        if(event_num == -1)
        {
                print_help();
                return -1;
        }

        sprintf(input_buf,"/dev/input/event%d",event_num);

        fd = open(input_buf, O_RDONLY);
        pfd.fd = fd;
        pfd.events = POLLIN | POLLRDNORM;

        if(fd < 0){
                printf("can't open %s\n",input_buf);
                return -1;
        }

        while (1) {
                if (poll(&pfd, 1, -1) <= 0)
                        continue;

                if (read(fd, &t, sizeof(t)) != sizeof(t))
                        continue;

                printf("type:%d, code:%d, value:%ld\n", t.type, t.code,
t.value);

                if (t.type == EV_KEY) {
                        printf("key %d %s\n", t.code,
                                        (t.value) ? "Pressed" :
"Released");
                        if (t.code == KEY_POWER && !t.value) {
                                while (read(fd, &t, sizeof(t)) ==
sizeof(t))
                                        ;
                                break;
                        }
                }
                else{
                        if (t.type == EV_ABS)
                        {
                                if (t.type == EV_ABS&& t.code == ABS_X) {
                                        x = t.value;
                                }
                                if (t.type == EV_ABS && t.code == ABS_Y) {
                                        y = t.value;
                                }
                         }
                         if (t.type == EV_SYN) {
                                 printf("(%4d %4d)\n",x,y);
```

```
                        }
                }
        }

        close(fd);

        return 0;
}

CONSOLE_CMD(input, NULL, input_test, CONSOLE_CMD_MODE_SELF, "input test, press power to exit test")
```

6. 模块调试方法

使用测试命令测试。

7. 常见问题

暂无