HCRTOS key-gpio userguide

1. 文档履历

版本号	日期	制/修订人	制/修订记录
1.0	2023.08.12	邱浩佳	新增文档说明

2. 概述

2.1 编写目的

介绍key-gpio的功能和指导开发

2.2 读者对象

软件开发工程师和技术支持工程师

3. 模块介绍

3.1 设备树配置

```
gpio_key {
        status = "okay";
        debounce = <20>; /* ms */
        pinmux-active = <PINPAD_T14 0 PINPAD_T15 0>;
        key-num = <2>;
        /*gpio valid_status key_value*/
        gpio-key-map = <PINPAD_T14 GPIO_ACTIVE_LOW 106>,
                       <PINPAD_T15 GPIO_ACTIVE_HIGH 145>;
};
```

pinmux-active：第一位代表要使用的引脚，0 表示配置成 GPIO，两个一组；

valid_status：**指按键按下去的电平值，**低电平为 GPIO_ACTIVE_LOW ； 高电平为 GPIO_ACTIVE_HIGH ；

debounce = <20>; 消抖时间（ms）；

3.2 menuconfig配置

在sdk根路径下输入：make menuconfig，根据下面路径选中watchdog模式。

```
There is no help available for this option.|
| Symbol: CONFIG_DRV_GPIO_KEY [=y]|
| Type : bool|
| Prompt: hc key gpio driver|
| Location:|
| -> Components|
| -> kernel (BR2_PACKAGE_KERNEL [=y])|
| -> Drivers|
| -> input event (CONFIG_DRV_INPUT [=y])|
| Defined at input:58|
| Depends on: BR2_PACKAGE_KERNEL [=y] && CONFIG_DRV_INPUT [=y] &&
CONFIG_DRV_GPIO [=y]
```

输入：make kernel-rebuild all，编译选中的驱动。

4. 模块接口说明

暂无。

5. 模块测试用例与Sample Code

参考程序路径：components/cmds/source/watchdog/watchdog_test.c

```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);

         }

         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

暂无;

7. 常见问题

暂无;