HICHIP ADC KEY使用文档说明5.0

1. 文档履历

版本号	日期	制/修订人	制/修订记录
1.0	2023.10.25	邱浩佳	新增adc key使用说明
2.0	2023.10.25	邱浩佳	新增hclinux adc key使用说明
3.0	2023.11.21	邱浩佳	更新设备树按键范围设置说明
4.0	2023.11.29	邱浩佳	新增adc查询电压的使用说明
5.0	2024.06.17	邱浩佳	更新说明

2. 概述

2.1 编写目的

介绍hcchip adc的使用文档说明；

2.2 读者对象

hcchip sdk开发工程师和技术支持工程师；

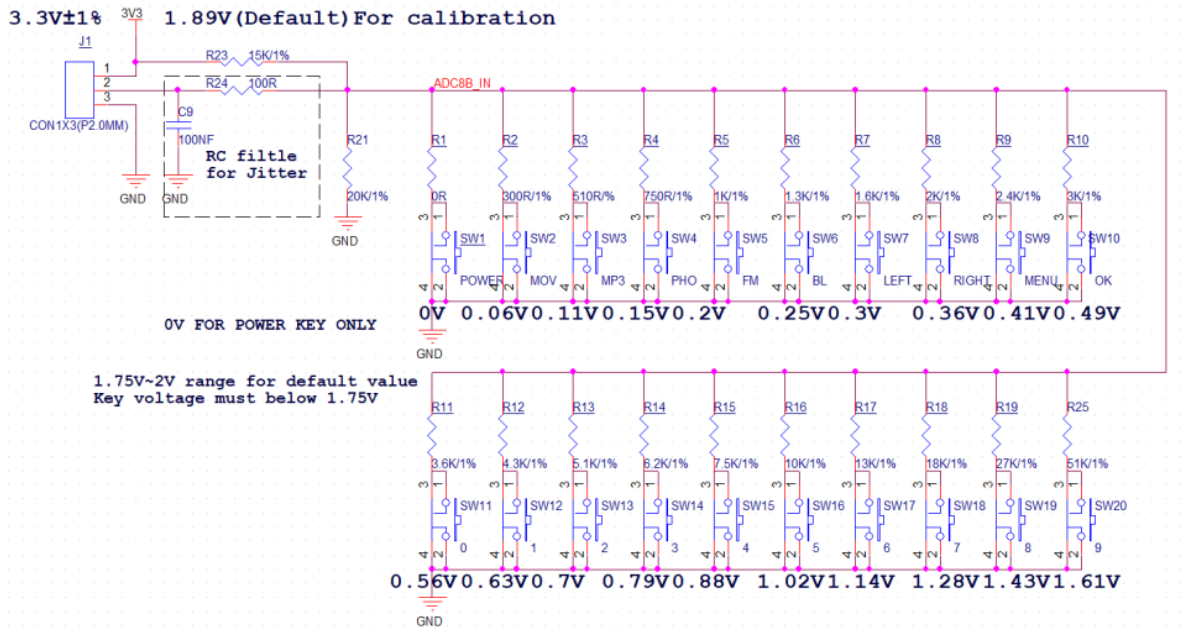
3. 模块介绍

- adc的电压的量程2V，超过2V的均显示为2V，超过3.3V存在烧毁芯片的风险；
- adc的精度为8bit；
- adc按键使用说明；
- adc查询电压的使用说明；

4. adc按键使用说明

4.1 硬件电路设计

每个按键之间要留有足够的余量，参见下面参考电路原理图，最高的按键电压不可以超过1.74V；



4.2 设备树节点添加

4.2.1 15xx

```
adc {
    devpath = "/dev/adc";
    status = "okay";
    adc_ref_voltage = <1890>;  /* voltage value(mV) when no any key pressed
*/
    key-num = <20>;
    /*key_map = <voltage_min voltage_max, code>*/
    key-map =<0 20 0x16>,<62 83 175>,
        <84 127 0x187>,<128 173 169>,
        <174 221 233>,<222 271 237>,
        <272 325 105>,<326 383 106>,
        <384 449 140>,<450 522 0x160>,
        <523 592 11>,<593 665 2>,
        <666 746 3>,<747 834 4>,
        <835 945 5>,<946 1074 6>,
        <1075 1205 7>,<1206 1352 8>,
        <1353 1520 9>,<1521 1650 10>;
        /* 当电压为1.74V时，参考按键电压范围最大值不应超过1830 */
};
```

4.2.2 16xx

4.2.2.1 hcrtos

```
key-adc@0 {
    status = "okay";
    adc_ref_voltage = <1890>; //range:0-2000mV /* voltage value(mV) when no any key pressed */
    key-num = <5>;
    key-map = <200 500 103>,
              <501 900 108>,
              <901 1100 105>,
              <1101 1300 106>,
              <1301 1740 0x160>;
};
```

4.2.2.2 hclinux

```
&key_adc0 {
        status = "okay";
        adc_ref_voltage = <1890>; //range:0-2000mV
        /*key_map = <voltage_min, voltage_max, keycode>*/
    key-map = <200 500 103>,
              <501 900 108>,
              <901 1100 105>,
              <1101 1300 106>,
              <1301 1740 0x160>;
};
```

4.3 menuconfig的配置

4.3.1 hcrtos

根据下面的路径打开adc按键的驱动

```
There is no help available for this option.
Symbol: CONFIG_KEY_ADC [=n]
Type  : bool
Prompt: key adc driver
  Location:
    -> Components
      -> kernel (BR2_PACKAGE_KERNEL [=y])
        -> Drivers
          -> input event (CONFIG_DRV_INPUT [=y])
            -> saradc menu (CONFIG_SAR_ADC [=y])
  Defined at ../saradc:1
  Depends on: BR2_PACKAGE_KERNEL [=y] && CONFIG_DRV_INPUT [=y] && CONFIG_SAR_ADC [=y]
  Selected by [n]:
  - CONFIG_STANDBY_ADC_WAKE_UP [=n] && BR2_PACKAGE_KERNEL [=y] && CONFIG_DRV_STANDBY [=y] && CONFIG_WAKE_UP_MODE [=y]
```

然后执行make kernel-rebuild all；

4.3.2 hclinux

根据下面的路径打开adc按键的驱动

```
There is no help available for this option.
Symbol: KEY_ADC [=y]
Type  : boolean
Prompt: key adc driver
  Location:
    -> Device Drivers
      -> HC drivers
        -> key adc driver (HC_ADC [=y])
  Defined at drivers/hcdrivers/adc/Kconfig:6
  Depends on: HC_ADC [=y]
```

然后执行make linux-rebuild all；

4.4 模块测试用例与Sample Code

参考代码位于sdk：hcrtos/components/cmds/source/input_event/input_test.c

```
"***********************************\n"
"input test cmds help\n");
"\tfor example : input_test -i 1\n");
"\t'i'   1 means event1\n");
"***********************************\n"

在串口控制台输入：input_test -i1，然后按下按键既可以获得到键值；
```

hclinux的测试方法可以用：hexdump /dev/input/event1，打开对应的按键输入设备既可以；

4.5 模块调试方法

设备树中按键电压的范围设置方式：可根据原理图设计电压，比如A键：800mV、B键1200mV；C键：1500mV；可将B范围设置为((800+1200)/2 (1200+1500)/2)即与相邻按键相加除以2；即B键的范围(1000 1350);

当按键个数相对较少时，可以根据实际情况扩大按键范围；

4.6 常见问题

Q：最高电压的按键无效；

A：最高按键的电压不应超过1.74V，超过1.74V会存在与不触发中断的范围发送冲突，造成失灵；

Q：最高电压的按键自己触发；

A：设备树中的最高电压按键的范围的最大值不应超过1830mV；

5. adc电压查询的使用

5.1 注意事项

15xx系列只有一路adc通道，16xx系列有6路adc通道；采集电压范围均为0-2V，采集精度均为8bit，超过2V均显示为0xff；超过3.3V会存在烧毁芯片的风险；

16xx系列最多有6路adc通道，具体数量请查看对应型号的芯片，如果直接使用adc通道测量电压会存在一定误差，可以用空闲通道在硬件上接1.89V电压为测量通道做校准；比如在硬件设计上使用通道0去测量电压，此时可以将通道1在硬件上接在1.89V，再在设备树里进行相应配置即可用软件校准通道0。

电压换算公式：驱动返回的值为寄存器值，需要进行转换为电压值；1890mV对应的寄存器值是241；V（测量值）= reg（测量值）* 1890 / 241;

5.2 如何使用

5.3 rtos如何使用adc查询电压

5.3.1 设备树配置

```
/* 15xx */
queryadc0 {
        devpath = "/dev/queryadc0";
        status = "okay";
};


/* 16xx */
queryadc0 {      //该节点表示adc通道0用来测量电压；
        //adjust_channel = <1>;     //adc通道1用来进行校准；
        devpath = "/dev/queryadc0";
        status = "okay";
};
```

5.3.2 menuconfig驱动勾选

```
There is no help available for this option.
Symbol: CONFIG_POLL_ADC [=y]
Type  : bool
Prompt: poll adc driver
  Location:
    -> Components
      -> kernel (BR2_PACKAGE_KERNEL [=y])
        -> Drivers
          -> input event (CONFIG_DRV_INPUT [=y])
            -> saradc menu (CONFIG_SAR_ADC [=y])
  Defined at ../saradc:5
  Depends on: BR2_PACKAGE_KERNEL [=y] && CONFIG_DRV_INPUT [=y] &&
CONFIG_SAR_ADC [=y]
```

编译命令：make kernel-rebuild all

5.3.3 测试命令以及测试代码

```
There is no help available for this option.
Symbol: CONFIG_CMDS_QUERY_ADC_TEST [=y]
Type  : bool
Prompt: query adc test cmds
  Location:
    -> Components
      -> Cmds (BR2_PACKAGE_CMDS [=y])
        -> adc test operations (CONFIG_CMDS_ADC_TEST [=y])
  Defined at saradc:9
  Depends on: BR2_PACKAGE_CMDS [=y] && CONFIG_CMDS_ADC_TEST [=y]
```

编译命令：make cmds-rebuild all

测试代码路径：components/cmds/source/saradc/query_adc_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>
#include <string.h>

#define BUF_SIZE 1024

static void print_help(void) {
        printf("************************************\n");
        printf("input test cmds help\n");
        printf("\tfor example : query_test -i1\n");
        printf("\t'i'   1 means queryadc1\n");
        printf("************************************\n");
}

```

```c
#define read_buf_len    16

static int queryadc_test(int argc, char *argv[])
{
        int fd, ret;
        char device_node[BUF_SIZE];
        char *s = "/dev/queryadc";
        unsigned char read_buf[read_buf_len];
        uint8_t sar_dout = 0;

        long tmp;
        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(device_node,"/dev/queryadc%d",event_num);

         fd = open(device_node, O_RDONLY);

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

         ret = read(fd, read_buf, sizeof(read_buf));
         printf("adc value is %d\n", read_buf[0]);
         printf("voltage value is %d mV\n", read_buf[0] * 2000 / 255);

         close(fd);

         return 0;
 }

 CONSOLE_CMD(queryadc, "adc_test", queryadc_test, CONSOLE_CMD_MODE_SELF,
"queryadc test")
```

5.4 linux如何使用adc查询电压

5.4.1 设备树配置

```
&check_adc0 {   //adc通道0用来测量电压；
        //adjust_channel = <1>; //adc通道1用来进行校准；
        status = "okay";
};
```

5.4.2 menuconfig的配置

```
There is no help available for this option.
Symbol: CHECK_ADC [=n]
Type  : boolean
Prompt: poll adc driver
  Location:
    -> Device Drivers
      -> HC drivers
        -> key adc driver (HC_ADC [=y])
  Defined at drivers/hcdrivers/adc/Kconfig:10
  Depends on: HC_ADC [=y]
```

编译命令：make linux-rebuild all

5.4.3 测试代码

代码位于sdk的路径：hclinux/SOURCE/linux-drivers/drivers/hcdrivers/adc/hc_adc_test.c；

```
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>

static const char *device = "/dev/check_adc3";

int main(int argc, char *argv[])
{
        int ret = 0;
        int fd;
        uint8_t buf;
        uint32_t val32;

        fd = open(device, O_RDWR);
        if (fd < 0) {
                printf("can't open device\n");
                return -1;
```

```
        }

        read(fd, &buf, 1);


        printf("read /dev/check_adc0 test:\n");
        printf("val:0x%x\n", buf);
        val32 = buf * 1890 / 241;
        printf("%d Vol= %d\n", __LINE__, val32);

        close(fd);

        return 0;
}
```

5.5 模块调试方法

暂无。

5.6 常见问题

暂无。