CMT4522

BSP_Button Application Note

Version 1.1

Author: HOPERF

Security: Public

Date: 2021.07

Copyright © 2020 Shenzhen Hope Microelectronics Co. Ltd. All rights reserved.
Reproduction in whole or in part is prohibited without the prior written permission of the copyright holder.

Revision History

Revision	Author	Date	Description
V1.0		2021.02.22	This document is used for 4520/4550/4522/4552.
V1.1		2021.07.20	Delete BSP_BTN_PRESS_RELEASE_ENABLE for low power consume.

2021.02.22

目录

图表目录

1 BSP_Button 是什么

BSP_Button 是一套按键中间件处理程序，硬件依赖于 GPIO、KSCAN，上层通过 TASK 调度方式实现对按键的监听，支持按键类型按下、长按开始、长按保持、释放等。

2 BSP_Button 基本配置

要使用 BSP_Button 需要做合适的硬件配置，并在 TASK 调度序列上添加 BSP_Button 的任务 Bsp_Btn_Init，并在 TASK 事件处理上添加 BSP_Button 的事件处理 Bsp_Btn_ProcessEvent。

2.1 文件说明

使用 BSP_Button 需要包含下面文件，并在应用 task 中添加硬件的初始化代码。

- bsp_gpio.c：GPIO 按键驱动，用 GPIO 按键时包含。
- kscan.c：KSCAN 按键驱动，用 KSCAN 按键时包含。
- bsp_button.c：BSP_Button 按键检测逻辑。
- bsp_button_task.c：BSP_Button 调度机制。

2.2 按键组成

目前按键数据类型是 BYTE，不同 bit 表示含义不同。

- 高 2 个 bit 表示按键类型，按下、长按开始、长按保持、释放等。
- 低 6 个 bit 表示按键值，取值范围是 0x00~0x3F。其中 0x30~0x3F 为系统预留，可以使用值为 0x00~0x2F。

2.3 按键分类

按键支持单独按键和组合按键。

- 单独按键：一个按键对应一个 GPIO 或一个 KSCAN 横列交叉点。
- 组合按键：多个单独按键组成组合按键，比如多个 GPIO 单独按键、多个 KSCAN 单独按键等组成组合按键。比如 P1 按下是单独按键，P2 按下是单独按键，P1P2 同时按下是组合按键。

组合按键依赖于单独按键。当使用组合按键时，单独按键在前，组合按键在后。比如一共 10 个按键，8 个单独按键，2 个组合按键，那么按键值 0~7 对应 8 个单独按键，按键值 8~9 对应 2 个组合按键。

2.4 硬件配置

根据需要选择合适的底层硬件。

- BSP_BTN_JUST_GPIO：只使用 GPIO 按键。
- BSP_BTN_JUST_KSCAN：只使用 KSCAN 按键。
- BSP_BTN_GPIO_AND_KSCAN：同时使用 GPIO 按键和 KSCAN 按键。

根据需要选择单独按键、组合按键的数量。

- BSP_SINGLE_BTN_NUM：单独按键数量。
- BSP_COMBINE_BTN_NUM：组合按键数量，如无可定义为 0。
- BSP_TOTAL_BTN_NUM：总按键数量，总按键数量等于单独按键数量加组合按键数量。

如果系统既有 GPIO 又有 KSCAN，那么需要分别指定各自类型的单独按键数量和组合按键数量。

- BSP_KSCAN_SINGLE_BTN_NUM：KSCAN 单独按键数量。
- BSP_GPIO_SINGLE_BTN_NUM：GPIO 单独按键数量。
- BSP_KSCAN_COMBINE_BTN_NUM：KSCAN 组合按键数量，如无可定义为 0。
- BSP_GPIO_COMBINE_BTN_NUM：GPIO 组合按键数量，如无可定义为 0。

如果系统既有 GPIO 又有 KSCAN，按键的排列方式是：KSCAN 单独按--GPIO 单独按键--组合按键。

2.5 应用示例

2.5.1 只有 GPIO

若只有 GPIO 按键，需要如下相关配置：

- bsp_button_task.h 指定硬件配置方式、按键数量等。
- bsp_gpio.h 中指定使用 GPIO 引脚数量、空闲电平状态。
- 应用程序中配置具体的引脚、回调函数等。如存在组合按键需要指定组按键的对应关系。

比如，有 3 个单独按键(P14、P15、P26) ，无按键时是高电平，2 个组合按键(P14P15、P14 P26)，回调函数是 hal_bsp_btn_callback。

按键对应关系：

按键值	对应硬件
0	P14 按下
1	P15 按下
2	P26 按下
3	P14P15 同时按下
4	P14P26 同时按下

表 1：GPIO 类型按键映射示例

参考代码：

```
bsp_button_task.h:
#define BSP_BTN_HARDWARE_CONFIG
BSP_BTN_JUST_GPIO
#define BSP_SINGLE_BTN_NUM
(GPIO_SINGLE_BTN_NUM)
#define BSP_COMBINE_BTN_NUM
(2)
#define BSP_TOTAL_BTN_NUM
(BSP_SINGLE_BTN_NUM +
BSP_COMBINE_BTN_NUM)

bsp_gpio.h:
#define GPIO_SINGLE_BTN_NUM
3
#define GPIO_SINGLE_BTN_IDLE_LEVEL
1

bsp_btn_demo.c:
uint32_t
usr_combine_btn_array[BSP_COMBINE_B
TN] =
{
    (BIT(0)|BIT(1)),
    (BIT(0)|BIT(2)),
};

void hal_bsp_btn_callback(uint8_t evt)
{
    LOG("gpio evt:0x%x    ",evt);
    switch(BSP_BTN_TYPE(evt))
    {
        case BSP_BTN_PD_TYPE:
            LOG("press down ");
            break;
        case BSP_BTN_UP_TYPE:
            LOG("press up ");
            break;
        case BSP_BTN_LPS_TYPE:
            LOG("long press start ");
            break;
        case BSP_BTN_LPK_TYPE:
            LOG("long press keep ");
            break;
        default:
```

```
                LOG("unexpected ");
                break;
        }

        LOG("value:%d\n",BSP_BTN_INDEX(evt));
}

Gpio_Btn_Info gpio_btn_info = {
        {P14,P15,P26},
        hal_bsp_btn_callback,
};

void Demo_Init( uint8 task_id )
{
        if(PPlus_SUCCESS == hal_gpio_btn_init(&gpio_btn_info))
        {
                bsp_btn_gpio_flag = TRUE;
        }
        else
        {
                LOG("hal_gpio_btn_init error:%d\n",__LINE__);
        }
}
```

2.5.2 只有 KSCAN

若只有 KSCAN 按键，需要如下相关配置：

- bsp_button_task.h 指定硬件配置方式、按键数量等。
- kscan.h 中指定使用 KSCAN 使用的行列数量。
- 应用程序中配置具体的引脚、回调函数等。如存在组合按键需要指定组按键的对应关系。

比如，有 4*4 个单独按键(ROW：P0、P2、P25、P18) (COL：P1、P3、P24、P20) ，2 个组合按键（Row0Col0 和 Row0Col1、Row0Col2 和 Row0Col3），回调函数是 hal_bsp_btn_callback。

按键值	对应硬件
0	Row0Col0 按下
1	Row0Col1 按下
2	Row0Col2 按下
3	Row0Col3 按下
4	Row1Col0 按下

5	Row1Col1 按下
6	Row1Col2 按下
7	Row1Col3 按下
8	Row2Col0 按下
9	Row2Col1 按下
10	Row2Col2 按下
11	Row2Col3 按下
12	Row3Col0 按下
13	Row3Col0 按下
14	Row3Col0 按下
15	Row3Col0 按下
16	Row0Col0、Row0Col1 同时按下
17	Row0Col2、Row0Col3 同时按下

表 2：KSCAN 类型按键映射示例

参考代码：

```
bsp_button_task.h:
#define BSP_BTN_HARDWARE_CONFIG     BSP_BTN_JUST_KSCAN
#define BSP_SINGLE_BTN_NUM          (NUM_KEY_ROWS * NUM_KEY_COLS)
#define BSP_COMBINE_BTN_NUM         (2)
#define BSP_TOTAL_BTN_NUM           (BSP_SINGLE_BTN_NUM + BSP_COMBINE_BTN_NUM)

kscan.h:
#define NUM_KEY_ROWS     4
#define NUM_KEY_COLS     4

bsp_btn_demo.c:
KSCAN_ROWS_e rows[NUM_KEY_ROWS] =
{KEY_ROW_P00,KEY_ROW_P02,KEY_ROW_P25,KEY_ROW_P18};
KSCAN_COLS_e cols[NUM_KEY_COLS] =
{KEY_COL_P01,KEY_COL_P03,KEY_COL_P24,KEY_COL_P20};

uint32_t usr_combine_btn_array[BSP_COMBINE_BTN_NUM] =
{
    (BIT(0)|BIT(1)),
    (BIT(2)|BIT(3)),
};

void hal_bsp_btn_callback(uint8_t evt)
```

```
{
    LOG("kscan evt:0x%x    ",evt);
    switch(BSP_BTN_TYPE(evt))
    {
        case BSP_BTN_PD_TYPE:
            LOG("press down ");
            break;
        case BSP_BTN_UP_TYPE:
            LOG("press up ");
            break;
        case BSP_BTN_LPS_TYPE:
            LOG("long press start ");
            break;
        case BSP_BTN_LPK_TYPE:
            LOG("long press keep ");
            break;
        default:
            LOG("unexpected ");
            break;
    }

    LOG("value:%d\n",BSP_BTN_INDEX(evt));
}

void Demo_Init( uint8 task_id )
{
    hal_kscan_btn_check(hal_bsp_btn_callback);
    if(bsp_btn_kscan_flag != TRUE)
    {
        LOG("hal_kscan_btn_check error:%d\n",__LINE__);
    }
}
```

2.5.3 GPIO 和 KSCAN 都有

若 GPIO 按键和 KSCAN 按键同时存在，需要分别配置各自使用的引脚和的引脚数量，如存在组合按键，需要指派组合按键关系。

- bsp_button_task.h 指定硬件配置方式、按键数量等。
- bsp_gpio.h 中指定使用 GPIO 引脚数量、空闲电平状态。
- kscan.h 中指定使用 KSCAN 使用的行列数量。
- 应用程序中配置具体的引脚、回调函数等。如存在组合按键需要指定组按键的对应关系。

比如，有 4*4 个 kscan 单独按键(ROW：P0、P2、P25、P18) (COL：P1、P3、P24、P20) 。有 3 个单独按键(P14、P15、P26) ，无按键时是高电平。1 个组合按键（row0col2、row0col3），1 个组合按键(P14P15)。回调函数是 hal_bsp_btn_callback。

按键值	对应硬件
0	Row0Col0 按下
1	Row0Col1 按下
2	Row0Col2 按下
3	Row0Col3 按下
4	Row1Col0 按下
5	Row1Col1 按下
6	Row1Col2 按下
7	Row1Col3 按下
8	Row2Col0 按下
9	Row2Col1 按下
10	Row2Col2 按下
11	Row2Col3 按下
12	Row3Col0 按下
13	Row3Col0 按下
14	Row3Col0 按下
15	Row3Col0 按下
16	P14 按下
17	P15 按下
18	P26 按下
19	Row0Col2、Row0Col3 同时按下
20	P14P15 按下

表 3：KSCAN 和 GPIO 类型按键映射示例

参考代码：

```
bsp_button_task.h:
#define BSP_BTN_HARDWARE_CONFIG       BSP_BTN_GPIO_AND_KSCAN

#define BSP_KSCAN_SINGLE_BTN_NUM      (NUM_KEY_ROWS * NUM_KEY_COLS)
```

```
#define BSP_GPIO_SINGLE_BTN_NUM         (GPIO_SINGLE_BTN_NUM)

#define BSP_KSCAN_COMBINE_BTN_NUM           (1)
#define BSP_GPIO_COMBINE_BTN_NUM            (1)

#define BSP_SINGLE_BTN_NUM    (BSP_KSCAN_SINGLE_BTN_NUM +\
                                            BSP_GPIO_SINGLE_BTN_NUM)
#define BSP_COMBINE_BTN_NUM       (BSP_KSCAN_COMBINE_BTN_NUM +\
                                            BSP_GPIO_COMBINE_BTN_NUM)
#define BSP_TOTAL_BTN_NUM      (BSP_SINGLE_BTN_NUM + \
                                            BSP_COMBINE_BTN_NUM)

kscan.h:
#define NUM_KEY_ROWS      4
#define NUM_KEY_COLS      4

bsp_gpio.h:
#define GPIO_SINGLE_BTN_NUM           3
#define GPIO_SINGLE_BTN_IDLE_LEVEL    1

bsp_btn_demo.c:
KSCAN_ROWS_e rows[NUM_KEY_ROWS] =
{KEY_ROW_P00,KEY_ROW_P02,KEY_ROW_P25,KEY_ROW_P18};
KSCAN_COLS_e cols[NUM_KEY_COLS] =
{KEY_COL_P01,KEY_COL_P03,KEY_COL_P24,KEY_COL_P20};

BTN_T usr_sum_btn_array[BSP_TOTAL_BTN_NUM];
uint32_t usr_combine_btn_array[BSP_COMBINE_BTN_NUM] =
{
      (BIT(2)|BIT(3)),
      (BIT(16)|BIT(17)),
};

void hal_bsp_btn_callback(uint8_t evt)
{
      LOG("kscan evt:0x%x    ",evt);
      switch(BSP_BTN_TYPE(evt))
      {
            case BSP_BTN_PD_TYPE:
                  LOG("press down ");
                  break;
            case BSP_BTN_UP_TYPE:
                  LOG("press up ");
                  break;
```

```
            case BSP_BTN_LPS_TYPE:
                LOG("long press start ");
                break;
            case BSP_BTN_LPK_TYPE:
                LOG("long press keep ");
                break;
            default:
                LOG("unexpected ");
                break;
        }

        LOG("value:%d\n",BSP_BTN_INDEX(evt));
}

Gpio_Btn_Info gpio_btn_info = {
        {P14,P15,P26},
        hal_bsp_btn_callback,
};

void Demo_Init( uint8 task_id )
{
        hal_kscan_btn_check(hal_bsp_btn_callback);
        if(bsp_btn_kscan_flag != TRUE)
        {
            LOG("hal_kscan_btn_check error:%d\n",__LINE__);
        }

        if(PPlus_SUCCESS == hal_gpio_btn_init(&gpio_btn_info))
        {
            bsp_btn_gpio_flag = TRUE;
        }
        else
        {
            LOG("hal_gpio_btn_init error:%d\n",__LINE__);
        }
}
```

3 BSP_Button 扩展配置

3.1 支持按键类型可配

默认只支持按键按下，如果需要支持按键释放和长按键需要配置。

- BSP_BTN_LONG_PRESS_ENABLE：定义后支持长按键，长按键包括长按开始和长按保持。

3.2 时间参数可配

下面四个时间相关参数可配：

- BTN_SYS_TICK：调到周期，单位是 ms。
- BTN_FILTER_TICK_COUNT：消抖时间， BTN_SYS_TICK* BTN_FILTER_TICK_COUNT。
- BTN_LONG_PRESS_START_TICK_COUNT：长按开始时间，BTN_SYS_TICK* BTN_LONG_PRESS_START_TICK_COUNT。
- BTN_LONG_PRESS_KEEP_TICK_COUNT：长按保持时间，BTN_SYS_TICK* BTN_LONG_PRESS_KEEP_TICK_COUNT。

3.3 支持按键数量

默认配置最多支持 48 个按键，如不够用，可根据需要修改代码。