

# HCRTOS SDK pwm userguide

## 1. 目录

HCRTOS SDK pwm userguide

- 1. 目录
- 2. 文档履历
- 3. 概述
  - 3.1 编写目的
  - 3.2 读者对象
- 4. 模块介绍
- 5. 模块接口说明
  - 5.1 接口函数
- 6. 模块测试用例与Sample Code
- 7. 模块调试方法
- 8. 常见问题

## 2. 文档履历

版本号	日期	制/修订人	制/修订记录
1.0	2023.04.10	邱浩佳	新增说明文档
2.0	2023.04.23	邱浩佳	支持在初始化的时候设置占空比和周期
3.0	2023.04.25	邱浩佳	取消在初始化时设置占空比和周期，以及在不同阶段避免多次初始化
4.0	2023.08.14	邱浩佳	新增极性设置
5.0	2023.09.09	邱浩佳	新增可配置pwm范围选择

## 3. 概述

## 3.1 编写目的

指导和介绍pwm的使用和开发

## 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

- rtos的pwm模块区分15xx系列和16xx系列的芯片，15xx系列的芯片有3路pwm通道；16xx系列的芯片有6路pwm通道
- 当前驱动下，pwm的最大clk为27M，27M时钟下最大频率的周期为37ns。
- 支持在驱动初始化时设置占空比和周期。

## 4.1 设备树配置

以pwm0为例子：

```
1 pwm0 {  
2     pinmux-active = <PINPAD_L23 1>;  
3     devpath = "/dev/pwm0";  
4     polarity = <1>;  
5     status = "okay";  
6 };
```

pinmux-active：将要设置的引脚PINPAD\_L23配置成pwm模式，并不是配置成pwm的值都写为1，具体的引脚配置成pwm请查看具体配置根据所使用引脚参考

components/kernel/source/include/uapi/hcuapi/pinmux/hc15xx\_pinmux.h和

components/kernel/source/include/uapi/hcuapi/pinmux/hc16xx\_pinmux.h。

devpath：生成的pwm节点的路径。

polarity：初始化完pwm后通道的极性。0：低电平；1：高电平；

status：状态为okay代表启用；disabled代表不启用。

**pwm的占空比和频率均通过代码设置，请参考sample code。**

## 4.2 menuconfig配置

在SDK根目录输入make menuconfig，安装下面路径进行选中pwm

```
1 Location:  
2     -> Components  
3     -> kernel (BR2_PACKAGE_KERNEL [=y])  
4     -> Drivers
```

```
----- Drivers -----
-----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <E>

[*] Uart driver --->
[ ] Virtual uart driver ----
[ ] File uart driver ----
[*] Dummy uart driver
[*] Enable /dev/null
[ ] Enable /dev/zero
-- Softirq support
-- Lnx support --->
[*] Workqueue support --->
[ ] AMP RPC support ----
[*] Ramdisk
[*] RAM disk wrapper (mkrd)
[*] Block-to-Character (BCH) Support --->
[*] Memory Technology Device (MTD) Support --->
[ ] FIFO and named pipe drivers ----
  Buffering --->
[*] mmz dev
[*] video --->
[*] pinmux
[*] gpio
[*] ADC Support
[*] Audio Support --->
[*] Video Support --->
[*] avsync dev Support
[ ] pok Support
[*] input event --->
[*] spi support --->
[*] lvds --->
[*] mipi
-- I2C Driver Support --->
[*] PWM Driver Support ----
[ ] NB debug
[*] audio platform support ----
[ ] watchdog ----
[*] persistent memory
-- hwspinlock
+ (+)
```

### 4.2.1 pwm频率范围选择

为了满足pwm拥有0-100%，每个可调节level为1%的需求，可以设置pwm范围内的值，但也可以设置超过范围最大值的频率，只是可调节的level不确定，根据所配置的频率变化。

当只使用一个通道时，可以配置CONFIG\_RANGE\_AUTO\_SET，分频系数会自动计算。，但超过该分频系数的范围时，可调节level不确定。

> Components > kernel > Drivers > PWM Driver Support

About How to calculated PWM frequency  
 $\text{freq} = 27\text{M} / (\text{div} * (\text{h\_val} + \text{l\_val}));$   
 $\text{h\_val} = \text{duty\_ns} / \text{div} / 37;$   
 $\text{l\_val} = (\text{period\_ns} - \text{duty\_ns}) / \text{div} / 37;$   
To meet the adjustable level of 1%,  
the h\_val and l\_val have following requirements:  
     $\text{Max}(\text{h\_val} + \text{l\_val}) = 65535;$   
     $\text{Min}(\text{h\_val} + \text{l\_val}) = 100;$   
the div can set in make menuconfig:  
    CONFIG\_RANGE\_411Hz\_270KHz: div = 1;  
    CONFIG\_RANGE\_206Hz\_135KHz: div = 2;  
    CONFIG\_RANGE\_103Hz\_67KHz : div = 4;  
    CONFIG\_RANGE\_50Hz\_33KHz : div = 8;  
    CONFIG\_RANGE\_AUTO\_SET : when use only one channel ,  
                                    div will auto calculated to meet with the frequency you set;  
if div = 1; the rang is 411(Hz)-270(KHz),  
you can set bigger than 270(KHz), but adjustable level may not be 1%;

Prompt: Select pwm frequency range  
Location:  
    -> Components  
        -> kernel (BR2\_PACKAGE\_KERNEL [=y])  
            -> Drivers  
                -> PWM Driver Support (CONFIG\_PWM\_DRIVER [=y])  
Defined at pwm:6  
Depends on: BR2\_PACKAGE\_KERNEL [=y] && CONFIG\_PWM\_DRIVER [=y]  
Selected by [m]:  
- BR2\_PACKAGE\_KERNEL [=y] && CONFIG\_PWM\_DRIVER [=y] && m

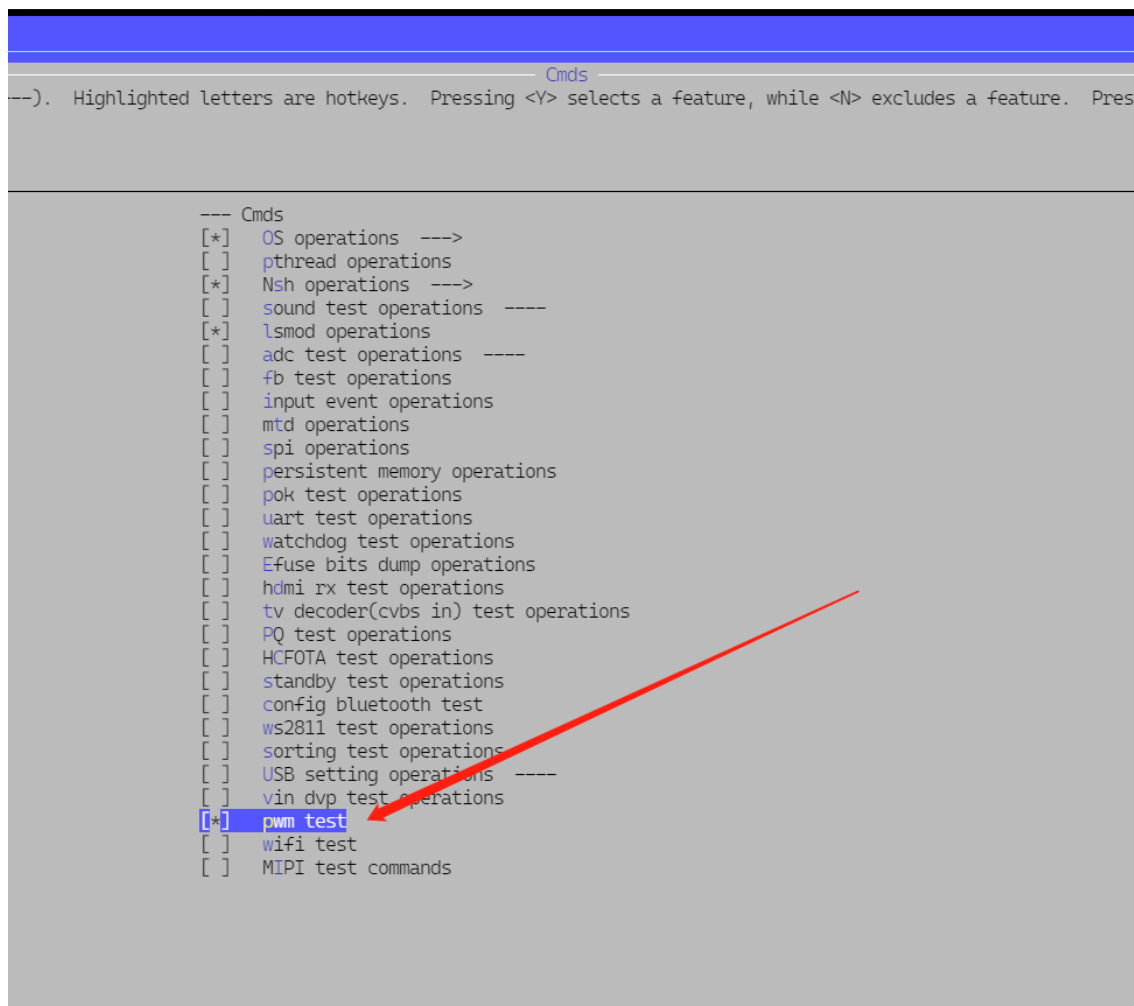
配置完成后输入make kernel-rebuild all，进行编译和烧录，在串口控制台输入下面命令既可以查看pwm节点。

```
hc1512a@dbB200#  
hc1512a@dbB200# nsh  
hc1512a@dbB200(nsh)# ls  
/:  
dev/  
hc1512a@dbB200(nsh)# cd dev  
hc1512a@dbB200(nsh)# ls  
/dev:  
auddec  
audsink  
avsync0  
avsync1  
bus/  
dis  
fb0  
ge  
input/  
mmz  
mtdblock0  
mtdblock1  
mtdblock2  
mtdblock3  
mtdblock4  
null  
persistentmem  
pwm0  
si_prodetect  
sndC0i2so  
uart1  
uart_dummy  
viddec  
vidsink  
hc1512a@dbB200(nsh)#
```

### 4.3 测试命令

在SDK根目录输入make menuconfig，根据下面路径选中pwm\_test命令。

```
1 Location:
2   -> Components
3   -> Cmds (BR2_PACKAGE_CMD5 [=y])
```



配置完成后，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使用测试命令。

```
hc1512a@dbB200#  
hc1512a@dbB200#  
hc1512a@dbB200# pwm_test -h  
Usage: pwm_test  
-s --start                start pwm device id  
-p --period_ns            set period_ns  
-d --duty_ns              set duty_ns  
-P --ploarity             set ploarity  
-S --stop                 stop pwm device id  
  
Invalid command "pwm_test"  
hc1512a@dbB200# pwm_test -s0 -p1000000 -d400000 -P0  
start /dev/pwm0 test  
hc1512a@dbB200#
```

## 5. 模块接口说明

## 5.1 接口函数

本模块提供ioctl函数接口供应用层使用。

- 设置pwm的参数，包括占空比、周期和极性。**注意占空比和周期的单位是ns。**

```
1 | info.polarity = polarity;
2 | info.period_ns = period_ns;
3 | info.duty_ns = duty_ns;
4 | ioctl(fd, PWMIOC_SETCHARACTERISTICS, (unsigned long)&info);
```

- 获取pwm的参数，包括占空比、周期和极性。**注意占空比和周期的单位是ns。**

```
1 | info.polarity = polarity;
2 | info.period_ns = period_ns;
3 | info.duty_ns = duty_ns;
4 | ioctl(fd, PWMIOC_GETCHARACTERISTICS, (unsigned long)&info);
```

- 开启pwm和关闭pwm。

```
1 | ioctl(fd, PWMIOC_START);
2 | ioctl(fd, PWMIOC_STOP);
```

## 6. 模块测试用例与Sample Code

代码存放路径为：components/cmds/source/pwm/pwm\_test.c。**注意占空比和周期的单位是ns。**

```
1 | #include <stdio.h>
2 | #include <unistd.h>
3 | #include <fcntl.h>
4 | #include <poll.h>
5 | #include <signal.h>
6 | #include <stdlib.h>
7 | #include <string.h>
8 | #include <getopt.h>
9 | #include <sys/ioctl.h>
10 | #include <hcuapi/pwm.h>
11 | #include <kernel/lib/console.h>
12 |
13 | static void print_usage(const char *prog)
14 | {
15 |     printf("Usage: %s \n", prog);
16 |     puts(" -s --start      start pwm device id\n"
17 |         " -p --period_ns   set period_ns\n"
18 |         " -d --duty_ns      set duty_ns\n"
19 |         " -P --ploarity     set ploarity\n"
20 |         " -S --stop        stop pwm device id\n");
21 | }
22 |
23 | static int pwm_test(int argc, char *argv[])
24 | {
```

```

25     int fd;
26     int id = 0;
27     bool stop_t = false;
28     char path[64] = { 0 };
29     struct pwm_info_s info = { 0 };
30     uint32_t period_ns, duty_ns, polarity;
31     period_ns = 1000000;
32     duty_ns = 500000;
33     polarity = 0;
34
35     opterr = 0;
36     optind = 0;
37
38     while (1) {
39         static const struct option lopts[] = {
40             { "start",          1, 0, 's' },
41             { "period_ns",      1, 0, 'p' },
42             { "duty_ns",        1, 0, 'd' },
43             { "stop",           1, 0, 's' },
44             { "polarity",       1, 0, 'P' },
45             { "NULL",          0, 0, 0 },
46         };
47
48         int c;
49
50         c = getopt_long(argc, argv, "s:p:d:S:P:", lopts, NULL);
51
52         if (c == -1) {
53             break;
54         }
55
56         switch(c) {
57             case 's':
58                 id = atoi(optarg);
59                 break;
60             case 'p':
61                 period_ns = atoi(optarg);
62                 break;
63             case 'P':
64                 polarity = atoi(optarg);
65                 break;
66             case 'd':
67                 duty_ns = atoi(optarg);
68                 break;
69             case 'S':
70                 stop_t = true;
71                 id = atoi(optarg);
72                 break;
73             default:
74                 print_usage(argv[0]);
75                 return -1;
76         }
77     }
78
79     sprintf(path, "/dev/pwm%d", id);
80     fd = open(path, O_RDWR);
81     if (fd < 0) {
82         printf("%s open fail\n", path);

```

```
83         return 0;
84     }
85
86     if (stop_t == true) {
87         printf("stop %s test\n", path);
88         ioctl(fd, PWMIOC_STOP);
89         close(fd);
90         return 0;
91     }
92
93     info.polarity = polarity;
94     info.period_ns = period_ns;
95     info.duty_ns = duty_ns;
96     ioctl(fd, PWMIOC_SETCHARACTERISTICS, (unsigned long)&info);
97
98     ioctl(fd, PWMIOC_START);
99     close(fd);
100    printf("start %s test\n", path);
101
102    return 0;
103 }
104
105 console_cmd(pwm_test, NULL, pwm_test, console_cmd_mode_self, "pwm test
operations")
```

## 7. 模块调试方法

---

可以用示波器或逻辑分析仪抓取波形查看。

## 8. 常见问题

---

暂无。