

# SigmaStar Camera PWM 使用参考



© 2019 SigmaStar Technology Corp. All rights reserved.

SigmaStar Technology makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by SigmaStar Technology arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

SigmaStar is a trademark of SigmaStar Technology Corp. Other trademarks or names herein are only for identification purposes only and owned by their respective owners.



{Product Description} {Document Name + Version}

# **REVISION HISTORY**

<b>Revision No.</b>	Description	Date
{000001}	• {Initial release}	{07/28/2018}



{Product Description} {Document Name + Version}

## **TABLE OF CONTENTS**

<b>REV</b>	REVISION HISTORY 错误!未定义书签。				
TAI	TABLE OF CONTENTS 错误!未定义书签。				
	概述				
	1.1. DEFAULT PWM PAD	错误!未定义书签。			
	LINUX PWM 控制				
	2.1. CONSOLE 下控制 PWM				



{Product Description} {Document Name + Version}

## 1. 概述

## 1.1. 概述

本文描述透过提供的硬件 PWM 如何产生特定的波型

## 1.1 Default PWM Pad

```
PWM 00 -> PAD_GPI08

PWM 01 -> PAD_GPI09

PWM 02 -> PAD_GPI010

PWM 03 -> PAD_GPI011

PWM 04 -> PAD_GPI012

PWM 05 -> PAD_GPI013

PWM 06 -> PAD_GPI014

PWM 07 -> PAD_GPI015

PWM 08 -> PAD_SD0_GPI00

PWM 09 -> PAD_FUART_CTS
```



{Product Description} {Document Name + Version}

## 2. LINUX PWM 控制

## 2.1. Console 下控制 PWM

1. Motor hierarchy

Group 0

PWM 0

PWM 1

PWM 2

PWM 3

Group 1

PWM 4

PWM 5

PWM 6

PWM 7

Group 2

PWM 8

PWM 9

**PWM 10** 

2. Cd 马达控制路径

Command:

cd /sys/devices/virtual/mstar/motor

- 3. Set mode/period(frequency) / Begin/End / round number/enable/hold/stop
  - mode

Command:

#### echo PWM\_ID enable > group\_mode

ex:echo 01 > group\_mode # 设定 PWM0 为马达模式

ex:echo 00> group\_mode # 取消 PWM0 为马达模式

period

Command:

#### echo PWM\_ID period > group\_period

In our driver implementation, xxxx indicates output frequency

ex: echo 0 2000 > group\_period # PWM0 will generate 2KHz waveform

• begin

Command:

echo PWM\_ID begin > group\_begin

ex: echo 0 10 > group\_begin # PWM0 will generate duty\_cycle starting from 10% of the period



{Product Description} {Document Name + Version}

end

Command:

#### echo PWM ID end > group end

ex: echo 0 25 > group\_end # PWM0 will generate duty\_cycle ending at 25% of the period

round

Command:

#### echo GROUP ID round > group round

ex: echo 0 10000 > group\_round # Group 0 will generate 10000 period of waveform.

enable

Command:

#### echo GROUP\_ID enable > group\_enable

ex: echo 0 1 > group\_enable # Group 0 start generating the waveform ex: echo 0 0 > group\_enable # Group 0 stop generating the waveform

hold

Command:

#### echo GROUP ID > group hold

ex: echo 0 > group\_hold # Group 0 hold the last complete waveform

stop

Command:

#### echo GROUP\_ID > group\_stop

ex: echo 0 > group\_stop # Group 0 immediately stop the waveform

duty\_qe0

Command:

## echo QE0\_EN > group\_duty\_qe0

ex: echo 0 > group\_duty\_qe0 # QE0 enable/disable

hold mode1

Command:

#### echo MODE1 EN > group hold mode1

ex: echo 1 > group\_hold\_mode1 # Mode1 enable

Mode 0 (keep pwm state)

在拉 hold 期间更改设定,且 pwm 值保持原始

Mode 1 (pull low)

在拉 hold 期间更改设定,但 pwm 值数完最后一个 period 后会拉 low



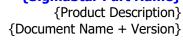
{Product Description} {Document Name + Version}

Hold 0 会有 interrupt, 这个 interrupt 是下了 hold 0 之后, 这个 period 结束, HW 会发 interrupt, 然后 driver 在 ISR 里头填新的参数, 取消 hold 0, 然后 PWM 就会产生新的参数的波型了

所以操作应该是这样 红色代表是 AP(客户) 的操作 蓝色代表是 Driver 的操作 绿色代表是 HW 的行为

#### 填参数 →enable 之后, 以下分成 hold 0/1 来解释

- 1. Hold 0
  - A. 下新的参数 (驱动先把新的参数放在内存中)  $\rightarrow$  打开 hold  $0 \rightarrow$  一个 period 结束/保持最后波型  $\rightarrow$  (ISR) 驱动从内存把参数回填 PWM/放开 hold  $0 \rightarrow$  产生新的波型
- 2. Hold 1
  - A. 打开  $Hold 1 \rightarrow \uparrow$  period 结束/波形拉低  $\rightarrow$  (AP 确定 period 结束) 下新的参数  $\rightarrow$  放开  $Hold 1 \rightarrow$  产生新的波形

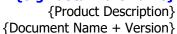




pwm enable(mstar pwm);

### 2.2.Kernel driver 控制 PWM

```
#include <linux/pwm.h>
#static struct pwm device *mstar pwm;
    struct pwm device *pwm request(int pwm id, const char *label)
       pwm_request - request a PWM device
   int pwm_config(struct pwm_device *pwm, int duty_ns, int period ns)
       pwm config - change a PWM device configuration
    int pwm enable(struct pwm device *pwm, int duty ns, int period ns)
       pwm_enable - start a PWM output toggling
sample:
      enable PWM0(PAD PWM0)
    static struct pwm device *mstar pwm;
       mstar pwm = pwm request(0, "MSTAR LCD");
       if (IS ERR(mstar pwm)) {
               printk("Start Unable to request PWM for LCD power!\n");
               return PTR ERR (mstar pwm);
       pwm config(mstar pwm, 50, 1000);//duty=50, hz=1000
```





## 3. UBOOT 标准 PWM 控制接口

#### **3.1.PWM API**

#### #include <pwm.h>

- int pwm init(id, div, invert)
  - sets up the clock speed and whether or not it is inverted
- int pwm config(id, duty, period ns)
  - sets up the duty and period
- int pwm\_enable(int pwm\_id)
  - enables the PWM driver
- int pwm disable(int pwm id)
  - disables the PWM driver

#### 参数说明:

• Id: 0:PWM0, 1:PWM1.....7:PWM7

DIV: 设置范围 0~65535Invert: 设置范围 0 或 1Duty: 设置范围 0~100

● Period: 设置范围 2~262143 (在这边的 Period 指的是 HZ)

{Product Description} {Document Name + Version}



# 3.2. PWM Reference Setting Table

PWM_DIV	PWM_PERIOD	FREQ.
(REG DIV+1)	(REG PERIOD+1)	(Hz)
1	2	6,000,000.00
1	4	3,000,000.00
1	8	1,500,000.00
1	16	750,000.00
1	32	375,000.00
1	64	187,500.00
1	128	93,750.00
1	256	46,875.00
1	512	23,437.50
1	1024	11,718.75
1	2048	5859.38
1	4096	2929.69
1	8192	1464.84
1	16384	732.42
1	32768	366.21
1	65536	183.10
1	131072	91.55
1	262144	45.78
2	262144	22.89
4	262144	11.44
8	262144	5.72
16	1024	2.86
32	262144	1.43
64	262144	0.715
128	262144	0.358
256	262144	0.1788

PWM_DIV (REG_DIV+1)	PWM_PERIOD (REG_PERIOD+1)	FREQ. (Hz)
512	262144	0.0894
1024	262144	0.0447
2048	262144	0.02235
4096	262144	0.01117
8192	262144	0.005588
16384	262144	0.002794
32768	262144	0.001397
65536	262144	0.000698