



SigmaStar Camera GPIO 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.



REVISION HISTORY

Revision No.	Description	Date
{0.1}	• {Initial release}	{07/28/2018}

TABLE OF CONTENTS

REVISION HISTORY	i
TABLE OF CONTENTS.....	ii
1. 概述.....	1
1.1. 概述.....	1
1.2. GPIO NUM 与 PAD 对应表	1
2. 内核使用 GPIO.....	2
2.1. 申请为 gpio 端口.....	2
2.2. 设为输入	2
2.3. 设为输出	3
2.4. 获取输入电平	3
2.5. 设置输出电平	3
3. 用户空间使用 GPIO	5
3.1. export/unexport 文件接口	5
3.2. /sys/class/gpio/gpioN.....	6
3.3. 示例.....	7
4. UBOOT 使用 GPIO	8
4.1. CMD: gpio -Config gpio port.....	8
4.2. API	8
4.2.1 设为输入	8
4.2.2 设为输出	8
4.2.3 获取输入电平.....	9
4.2.4 设置输出高电平	9
4.2.5 设置输出低电平	9

1. 概述

1.1. 概述

GPIO 采用标准的 LINUX 框架，能够使用统一的接口来操作 gpio。

1.2. GPIO NUM 与 PAD 对应表

请查看硬件线路图上 GPIO 的 PAD name，查找这个表格，对应的 num 就是你要操作的 GPIO 的 num。

例如：硬件线路图上的一个 GPIO 是 PAD_PM_GPIO8，如果要操作这个 GPIO，对应的 num =129。

开放客户使用的 GPIO 列表：

Table 1: 表 1-1

PAD_GPIO0	0	PAD_SR_IO06	28	PAD_SD_D0	56	PAD_ETH_TN	84
PAD_GPIO1	1	PAD_SR_IO07	29	PAD_SD_D1	57	PAD_ETH_TP	85
PAD_GPIO2	2	PAD_SR_IO08	30	PAD_SD_D2	58	PAD_USB_DM	86
PAD_GPIO3	3	PAD_SR_IO09	31	PAD_SD_D3	59	PAD_USB_DP	87
PAD_GPIO4	4	PAD_SR_IO10	32	PAD_PM_SD_CDZ	60	PAD_SD1_I00	88
PAD_GPIO5	5	PAD_SR_IO11	33	PAD_PM_IRIN	61	PAD_SD1_I01	89
PAD_GPIO6	6	PAD_SR_IO12	34	PAD_PM_GPIO0	62	PAD_SD1_I02	90
PAD_GPIO7	7	PAD_SR_IO13	35	PAD_PM_GPIO1	63	PAD_SD1_I03	91
PAD_GPIO8	8	PAD_SR_IO14	36	PAD_PM_GPIO2	64	PAD_SD1_I04	92
PAD_GPIO9	9	PAD_SR_IO15	37	PAD_PM_GPIO3	65	PAD_SD1_I05	93
PAD_GPIO12	10	PAD_SR_IO16	38	PAD_PM_GPIO4	66	PAD_SD1_I06	94
PAD_GPIO13	11	PAD_SR_IO17	39	PAD_PM_GPIO7	67	PAD_SD1_I07	95
PAD_GPIO14	12	PAD_UART0_RX	40	PAD_PM_GPIO8	68	PAD_SD1_I08	96
PAD_GPIO15	13	PAD_UART0_TX	41	PAD_PM_GPIO9	69		
PAD_FUART_RX	14	PAD_UART1_RX	42	PAD_PM_SPI_CZ	70		
PAD_FUART_TX	15	PAD_UART1_TX	43	PAD_PM_SPI_CK	71		
PAD_FUART_CTS	16	PAD_SPI0_CZ	44	PAD_PM_SPI_DI	72		
PAD_FUART_RTS	17	PAD_SPI0_CK	45	PAD_PM_SPI_DO	73		
PAD_I2C0_SCL	18	PAD_SPI0_DI	46	PAD_PM_SPI_WPZ	74		
PAD_I2C0_SDA	19	PAD_SPI0_DO	47	PAD_PM_SPI_HLD	75		
PAD_I2C1_SCL	20	PAD_SPI1_CZ	48	PAD_PM_LED0	76		
PAD_I2C1_SDA	21	PAD_SPI1_CK	49	PAD_PM_LED1	77		
PAD_SR_IO00	22	PAD_SPI1_DI	50	PAD_SAR_GPIO0	78		
PAD_SR_IO01	23	PAD_SPI1_DO	51	PAD_SAR_GPIO1	79		
PAD_SR_IO02	24	PAD_PWM0	52	PAD_SAR_GPIO2	80		
PAD_SR_IO03	25	PAD_PWM1	53	PAD_SAR_GPIO3	81		
PAD_SR_IO04	26	PAD_SD_CLK	54	PAD_ETH_RN	82		
PAD_SR_IO05	27	PAD_SD_CMD	55	PAD_ETH_RP	83		

2. 内核使用 GPIO

2.1. 申请为 gpio 端口

【目的】

创建端口为 GPIO。

【语法】

```
int gpio_request(unsigned gpio, const char *label)
```

【参数】

Table 2: 表 1-1

参数名称	描述
gpio	Gpio num
label	具体名称

【返回值】

Table 2: 表 1-2

返回值	描述
0	成功。
Other	失败。

2.2. 设为输入

【目的】

标记 gpio 为输入。

【语法】

```
int gpio_direction_input(unsigned gpio);
```

【参数】

Table 3: 表 2-1

参数名称	描述
gpio	Gpio num

【返回值】

Table 4: 表 2-2

返回值	描述
0	成功。
Other	失败。

2.3. 设为输出

【目的】

标记 gpio 为输出。

【语法】

```
int gpio_direction_output(unsigned gpio, int value);
```

【参数】

Table 5: 表 2-3

参数名称	描述
gpio	Gpio num
value	输出值

【返回值】

Table 6: 表 2-4

返回值	描述
0	成功。
Other	失败。

2.4. 获取输入电平

【目的】

获取输入引脚的电平。

【语法】

```
int gpio_get_value(unsigned gpio);
```

【参数】

Table 7: 表 2-5

参数名称	描述
gpio	Gpio num

【返回值】

Table 8: 表 2-6

返回值	描述
Int	电平值

2.5. 设置输出电平

【目的】

设定输出引脚的电平。

【语法】

```
void gpio_set_value(unsigned gpio, int value);
```

【参数】



Table 9: 表 2-7

参数名称	描述
gpio	Gpio num
value	输出值

【返回值】

Table 10: 表 2-8

返回值	描述
0	成功。
Other	失败。

3. 用户空间使用 GPIO

用户空间访问 gpio，即通过 sysfs 接口访问 gpio

下面是/sys/class/gpio 目录下的三种文件：

--export/unexport 文件

--gpioN 指代具体的 gpio 引脚

--gpio_chipN 指代 gpio 控制器

必须知道以上接口没有标准 device 文件和它们的链接。

```
/sys/class/gpio # ls
export
gpiochip0
unexport
/sys/class/gpio #
```

Figure 3: 图 3-1

3.1. export/unexport 文件接口

/sys/class/gpio/export，该接口只能写不能读。

用户程序通过写入 gpio 的编号来向内核申请将某个 gpio 的控制权导出到用户空间，前提是没有内核代码申请这个 gpio 端口，如用户申请编号为 12 的 GPIO 的命令：

echo 12 > export

上述操作会为 12 号 gpio 创建一个节点 gpio12，此时/sys/class/gpio 目录下边生成一个 gpio12 的目录，如下图所示：

```
/sys/class/gpio #
/sys/class/gpio # echo 12 > export
[ 7067.555358] [GPIO][00134] [mstar-gpio]mstar_gpio_request offset=12
[ 7067.561649] [GPIO][00180] [mstar-gpio]mstar_gpio_to_irq,but not set reg
/sys/class/gpio # ls
export
gpio12
gpiochip0
unexport
/sys/class/gpio #
```

Figure 2: 图 3-2

/sys/class/gpio/unexport 和导出的效果相反，比如移除 gpio12 这个节点操作命令：

echo 12 > unexport

上述操作将会移除 gpio12 这个节点，如下图所示：

```
/sys/class/gpio # echo 12 > unexport
[ 7246.139183] [GPIO][00140] [mstar-gpio]mstar_gpio_free
/sys/class/gpio #
/sys/class/gpio #
/sys/class/gpio #
/sys/class/gpio # ls
export
gpiochip0
unexport
/sys/class/gpio #
/sys/class/gpio #
```

Figure 3: 图 3-3

3.2. /sys/class/gpio/gpioN

指代某个具体的 gpio 端口,里边有如下属性文件:

direction 表示 gpio 端口的方向,读取结果是 in 或 out。也可以对该文件进行写操作,写入 out 时该 gpio 设为输出同时电平默认为低。写入 low 或 high 时不仅可以设置为输出还可以设置指定的输出电平。当然如果内核不支持或者内核代码不愿意,将不会存在这个属性,比如内核调用了 gpio_export(N,0)就表示内核不愿意修改 gpio 端口方向属性。

value 表示 gpio 引脚的电平, 0 表示低电平, 1 表示高电平; 如果 gpio 被配置为输出, 这个值是可写的, 记住任何非零的值都将输出为高电平。如果某个引脚被配置为中断, 则可以调用 poll(2)函数监听该中断, 中断触发后 poll(2)函数就会返回。

```
/sys/class/gpio # echo 12 > export
[ 7446.305126] [GPIO][00134] [mstar-gpio]mstar_gpio_request offset=12
[ 7446.311406] [GPIO][00180] [mstar-gpio]mstar_gpio_to_irq,but not set reg
/sys/class/gpio #
/sys/class/gpio # ls
export
gpio12
gpiochip0
unexport
/sys/class/gpio # cd gpio12/
/sys/devices/virtual/gpio/gpio12 # ls
active_low
direction
edge
power
subsystem
uevent
value
/sys/devices/virtual/gpio/gpio12 # echo out > direction
[ 7496.197243] [GPIO][00173] [mstar-gpio]mstar_gpio_direction_output
/sys/devices/virtual/gpio/gpio12 # echo 1 > value
[ 7512.375372] [GPIO][00149] [mstar-gpio]mstar_gpio_set
/sys/devices/virtual/gpio/gpio12 # echo 0 > value
[ 7518.275654] [GPIO][00]mstar_gpio_set

/sys/devices/virtual/gpio/gpio12 # echo 1 > value
[ 7526.954376] [GPIO][00149] [mstar-gpio]mstar_gpio_set
/sys/devices/virtual/gpio/gpio12 #
```

Figure 4: 图 3-4



3.3. 示例

非中断模式

读操作: `int getGpioValue(int port)`, 传入要读去掉端口号, 返回该端口值 (0 或 1) ;

写操作: `void setGpioValue(int port, int value)`, 传入要设置的端口号和值。

4. UBOOT 使用 GPIO

4.1. CMD: gpio -Config gpio port

Usage:

gpio (for 2nd parameter, you must type at least 3 characters)

gpio output <gpio#> <1/0> : ex: gpio output 69 1

gpio input/get <gpio#> : ex: gpio input 10 (gpio 10 set as input)

gpio toggle <gpio#> : ex: gpio tog 49 (toggle)

gpio state <gpio#> : ex: gpio sta 49 (get i/o status(direction) & pin status)

gpio list [num_of_pins] : ex: gpio list 10 (list GPIO1~GPIO10 status)

4.2. API

4.2.1 设为输入

【目的】

标记 gpio 为输入。

【语法】

```
void MDrv_GPIO_Pad_Odn(MS_GPIO_NUM u32IndexGPIO);
```

【参数】

Table 11: 表 4-1

参数名称	描述
u32IndexGPIO	Gpio num

【返回值】

Table 12: 表 4-2

返回值	描述
void	

4.2.2 设为输出

【目的】

标记 gpio 为输出。

【语法】

```
void MDrv_GPIO_Pad_Oen(MS_GPIO_NUM u32IndexGPIO);
```

【参数】

Table 13: 表 4-3

参数名称	描述
u32IndexGPIO	Gpio num

【返回值】

Table 14: 表 4-4

返回值	描述
Void	

4.2.3 获取输入电平

【目的】

获取输入引脚的电平。

【语法】

```
U8 MDrv_GPIO_Pad_Read(MS_GPIO_NUM u32IndexGPIO);
```

【参数】

Table 15: 表 4-5

参数名称	描述
u32IndexGPIO	Gpio num

【返回值】

Table 16: 表 4-6

返回值	描述
unsigned char	电平值

4.2.4 设置输出高电平

【目的】

设定该引脚为高电平。

【语法】

```
void MDrv_GPIO_Pull_High(MS_GPIO_NUM u32IndexGPIO);
```

【参数】

Table 17: 表 4-7

参数名称	描述
gpio	Gpio num

4.2.5 设置输出低电平

【目的】

设定该引脚为低电平。

【语法】

```
void MDrv_GPIO_Pull_Low(MS_GPIO_NUM u32IndexGPIO);
```

【参数】

Table 18: 表 4-8



参数名称	描述
gpio	Gpio num