SigmaStar Camera GPIO User Guide

Version 0.1

REVISION HISTORY

Revision No.	Description	Date
0.1	• Initial release	12/18/2019

TABLE OF CONTENTS

1. OVERVIEW

1.1. General Description

The GPIO module is implemented using standard Linux architecture and is therefore operable via standard GPIO interface.

1.2. GPIO Number and Pad Mapping Table

Find the pad name of the GPIO from the hardware circuit diagram, and then check the following mapping table. The number corresponding to the pad name concerned represents the number of the GPIO you are using.
For example, suppose the GPIO shown in the hardware circuit diagram is PAD_PM_GPIO8, you should operate GPIO number 68 to use this GPIO. The GPIOs applicable for use are as listed below:

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_IO0	88
PAD_GPIO5	5	PAD_SR_IO11	33	PAD_PM_IRIN	61	PAD_SD1_IO1	89
PAD_GPIO6	6	PAD_SR_IO12	34	PAD_PM_GPIO0	62	PAD_SD1_IO2	90
PAD_GPIO7	7	PAD_SR_IO13	35	PAD_PM_GPIO1	63	PAD_SD1_IO3	91
PAD_GPIO8	8	PAD_SR_IO14	36	PAD_PM_GPIO2	64	PAD_SD1_IO4	92
PAD_GPIO9	9	PAD_SR_IO15	37	PAD_PM_GPIO3	65	PAD_SD1_IO5	93
PAD_GPIO12	10	PAD_SR_IO16	38	PAD_PM_GPIO4	66	PAD_SD1_IO6	94
PAD_GPIO13	11	PAD_SR_IO17	39	PAD_PM_GPIO7	67	PAD_SD1_IO7	95
PAD_GPIO14	12	PAD_UART0_RX	40	PAD_PM_GPIO8	68	PAD_SD1_IO8	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. USING GPIO INSIDE KERNEL

2.1. Creating a GPIO Port

Purpose:

Create a GPIO port.

Syntax:

int gpio_request(unsigned gpio, const char *label)

Parameter:

Parameter Name	Description
gpio	GPIO number
label	Number of the GPIO

Return Value:

Parameter Name	Description
0	Successful
Other	Failed

2.2. Setting a GPIO as Input

Purpose:

Set a GPIO as an input port.

Syntax:

int gpio_direction_input(unsigned gpio);

Parameter:

Parameter Name	Description
gpio	GPIO number

Return Value:

Parameter Name	Description
0	Successful
Other	Failed

2.3. Setting a GPIO as Output

Purpose:

Set a GPIO as an output port.

Syntax:

int gpio_direction_output(unsigned gpio, int value);

Parameter:

Parameter Name	Description
gpio	GPIO number
value	Output value

Return Value:

Parameter Name	Description
0	Successful
Other	Failed

2.4. Getting Input Level

Purpose:

Get the input level of an input pin.

Syntax:

int gpio_get_value(unsigned gpio);

Parameter:

Parameter Name	Description
gpio	GPIO number

Return Value:

Parameter Name	Description
Int	Level value

2.5. Setting Output Level

Purpose:

Set the level of an output pin.

Syntax:

void gpio_set_value(unsigned gpio, int value);

Parameter:

Parameter Name	Description
gpio	GPIO number
value	Output value

Return Value:

Parameter Name	Description
0	Successful
Other	Failed

3. USING GPIO IN USER SPACE

In this chapter, we will describe the access of GPIO from user space, that is, via the sysfs interface.
There are three files under the directory /sys/class/gpio, as follows:

- export/unexport file
- gpioN, which refers to a specific GPIO pin
- gpio_chipN, which refers to a specific GPIO controller

User should make sure there is no standard device file in this interface and be aware of the connection.

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

3.1. Exporting/Unexporting File Interface

The /sys/class/gpio/export interface is write-only and non-readable.
To use GPIO in user space, user program should write a specific GPIO number to kernel to request exportation of access right of a certain GPIO to user space, provided that no application for exportation of access right against this GPIO interface is under processing by the kernel. Take GPIO numbered 12 as an example. You should enter the command:

echo 12 > export

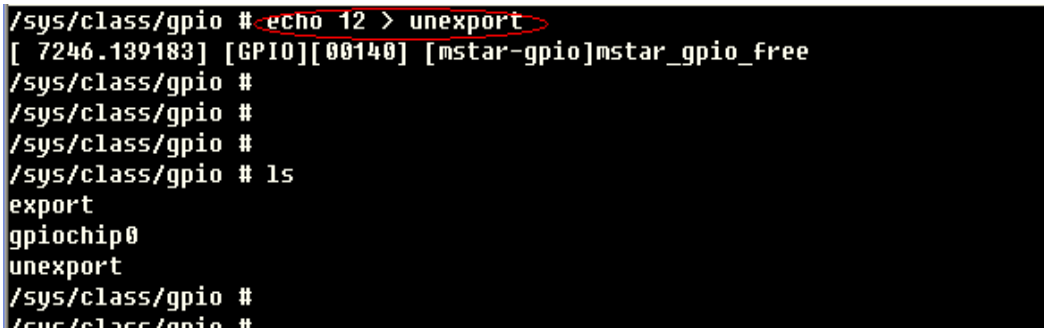
By this command, a node GPIO12 will be created for GPIO12, and a GPIO12 directory created under /sys/class/gpio, as shown below:

```
/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 #
```

Contrary to the export function, /sys/class/gpio/unexport allows you to remove the exported access right. To remove GPIO12, for example, enter the following command:

echo 12 > unexport

By this command, the GPIO12 node will be removed, as illustrated below:



3.2. /sys/class/gpio/gpioN

/sys/class/gpio/gpioN refers to a specific GPIO port, wherein the following attribute files are included:

direction, which indicates the direction of the GPIO port, that is, whether the read result is in or out. You can also do write operation against this file. For write out, the GPIO will be set as output with low level. In case of low write or high write, you can set the GPIO as output and besides specify the output level. Of course, if unsupported by kernel or declined by kernel code, this attribute will not exist. For example, if kernel calls gpio_export(N,0), it means the kernel does not want to modify the direction of the GPIO port.

value, which represents the level of the GPIO pin. 0 means low level, and 1 high level. If the GPIO is configured as an output, this value will be writeable. Note that all values other than 0 will be regarded as high level output. In case any GPIO pin is configured as an interrupt pin, you can call poll(2) function to monitor the interrupt status. Once the interrupt is triggered, the poll(2) function will be returned.

```
/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/qpio12 # 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 #
```

3.3. Example

In non-interrupt mode:

For read operation:

int getGpioValue(int port)

send the number of the GPIO port to be unexported, return value is 0 or 1.

For write operation:

void setGpioValue(int port, int value)

send the number and value of the GPIO port to be exported.

4. USING GPIO FOR UBOOT

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 Setting a GPIO as Input

Purpose:

Set a GPIO as an input port.

Syntax:

void MDrv_GPIO_Pad_Odn(MS_GPIO_NUM u32IndexGPIO);

Parameter:

Parameter Name	Description
u32IndexGPIO	GPIO number

Return Value:

Parameter Name	Description
void	

4.2.2 Setting a GPIO as Output

Purpose:

Set a GPIO as an output port.

Syntax:

void MDrv_GPIO_Pad_Oen(MS_GPIO_NUM u32IndexGPIO);

Parameter:

Parameter Name	Description
u32IndexGPIO	GPIO number

Return Value:

Parameter Name	Description
void	

4.2.3 Getting Input Level

Purpose:

Get the input level of an input pin.

Syntax:

U8 MDrv_GPIO_Pad_Read(MS_GPIO_NUM u32IndexGPIO);

Parameter:

Parameter Name	Description
u32IndexGPIO	GPIO number

Return Value:

Parameter Name	Description
unsigned char	Level value

4.2.4 Setting Output High Level

Purpose:

Set the level of an output pin as high level.

Syntax:

void MDrv_GPIO_Pull_High(MS_GPIO_NUM u32IndexGPIO);

Parameter:

Parameter Name	Description
gpio	GPIO number

4.2.5 Setting Output Low Level

Purpose:

Set the level of an output pin as low level.

Syntax:

void MDrv_GPIO_Pull_Low(MS_GPIO_NUM u32IndexGPIO);

Parameter:

Parameter Name	Description
gpio	GPIO number