在实际的驱动开发中，根据板级资源和CPU手册，GPIO初始化一般需要以下三个步骤：

       1.设置IO口的复用模式，如果某个IO当作GPIO使用，那么就需要根据CPU手册去配置iomux（IO复用寄存器）为GPIO模式；

        2.设置IO口的输入输出方向，根据实际开发需求，将相应的GPIO配置为相应的输入输出方向；

        3.GPIO初始化赋值（输出高低电平）、拉高拉低操作；

## 2.1 GPIO申请

#########################################

#description:申请一个GPIO资源

#unsigned gpio:要申请的GPIO管脚号，为一个正整数

# const char \*label：为申请的GPIO管脚取个名字

#########################################

**int gpio\_request(unsigned gpio, const char \*label);**

**int gpio\_request\_one(unsigned gpio, unsigned long flags, const char \*label);**

## 2.2 GPIO输入输出设置

#########################################

#description:设置某个GPIO的输入输出方向

#unsigned gpio:要设置的GPIO管脚号，为一个正整数

# int value：设置的值

#########################################

int gpio\_direction\_input(unsigned gpio);   
int gpio\_direction\_output(unsigned gpio, int value);

## 2.3 获取GPIO管脚的值和设置GPIO管脚的值

#########################################

#description:获取、设置某个GPIO的值

#unsigned gpio:要获取、设置的GPIO管脚号

# int value：设置的值

#########################################

int gpio\_get\_value(unsigned gpio);  
void gpio\_set\_value(unsigned gpio, int value);

## 2.4 GPIO当作中断口使用

#########################################

#description:设置某个GPIO为中断口

#unsigned gpio:要设置中断的GPIO管脚号

int gpio\_to\_irq(unsigned gpio);

返回的值即中断编号可以传给request\_irq()和free\_irq(),内核通过调用该函数将gpio端口转换为中断，在用户空间也有类似方法。

# 3. GPIO驱动实例

## 3.1 以下GPIO驱动例子为矩阵键盘中对GPIO的操作

//设置某个管脚为输入

int set\_key\_input(unsigned int gpio)

1. {
2. char name[32];
3. sprintf(name, "GPIO%d", gpio);
5. if(gpio\_request(gpio,NULL) != 0)
6. {
7. printk("gpio request error!\n");
8. return -1;
9. }
10. gpio\_direction\_input(gpio);
11. gpio\_free(gpio);  //释放资源
12. return 0;
13. }

int set\_key\_input(unsigned int gpio)

{

char name[32];

sprintf(name, "GPIO%d", gpio);

if(gpio\_request(gpio,NULL) != 0)

{

printk("gpio request error!\n");

return -1;

}

gpio\_direction\_input(gpio);

gpio\_free(gpio);

return 0;

}

//设置某个管脚为输出

1. int set\_key\_output(unsigned int gpio,int value)
2. {
3. char name[32];
4. sprintf(name, "GPIO%d", gpio);
6. if(gpio\_request(gpio,NULL) != 0)
7. {
8. printk("gpio request error!\n");
9. return -1;
10. }
11. gpio\_direction\_output(gpio,value);
12. gpio\_free(gpio);
13. return 0;
14. }

int set\_key\_output(unsigned int gpio,int value)

{

char name[32];

sprintf(name, "GPIO%d", gpio);

if(gpio\_request(gpio,NULL) != 0)

{

printk("gpio request error!\n");

return -1;

}

gpio\_direction\_output(gpio,value);

gpio\_free(gpio);

return 0;

}

//获取某个GPIO管脚的值

1. int get\_key\_value(unsigned int gpio)
2. {
3. int value= -1;
4. if (gpio\_request(gpio, NULL) != 0)
5. {
6. printk("get\_key\_value err\n");
7. return -1;
8. }
9. value = gpio\_get\_value(gpio);
10. gpio\_free(gpio);
12. return value;
13. }

int get\_key\_value(unsigned int gpio)

{

int value= -1;

if (gpio\_request(gpio, NULL) != 0)

{

printk("get\_key\_value err\n");

return -1;

}

value = gpio\_get\_value(gpio);

gpio\_free(gpio);

return value;

}

//设置某个GPIO输出为低电平

1. int set\_key\_low(unsigned int gpio)
2. {
4. if (gpio\_request(gpio, NULL) != 0) {
5. //printk("set\_key\_low request err\n");
6. return -1;
7. }
8. gpio\_direction\_output(gpio, 0);
9. gpio\_set\_value(gpio, 0);
10. gpio\_free(gpio);
12. return 0;
13. }

int set\_key\_low(unsigned int gpio)

{

if (gpio\_request(gpio, NULL) != 0) {

//printk("set\_key\_low request err\n");

return -1;

}

gpio\_direction\_output(gpio, 0);

\_\_gpio\_set\_value(gpio, 0);

gpio\_free(gpio);

return 0;

}

//拉高、拉低某个GPIO

//GPIO的拉高拉低操作内核没有提供通用的接口函数，这个需要驱动开发人员根据CPU手册的寄存器配置去封装拉高拉低函数，以下给出一个伪代码的例子：

//假设拉高GPIO1\_IO01这个IO：

1. <span style="font-size:12px;">#define SET\_PULL\_UP 0x01
2. #define SET\_PULL\_DOWN 0x00
3. #define REG\_GPIO\_BASE 0x8e000000
4. #define GPIO1\_IO01\_OFFSET 0x400
6. int set\_pull\_up(unsigned int reg\_base，unsigned int offset,int up)
7. {
8. unsigned int gpio\_base;
9. unsigned int gpio;
11. gpio\_base = ioreamap(reg\_base,SIZE\_4K);//调用ioreamap映射GPIO空间到内存，映射大小根据实际需求而定
12. gpio = gpio\_base + offset;
13. gpio |= up;        //将某个GPIO拉高，根据具体的寄存器操作而定
14. \_\_raw\_writel(gpio,gpio\_base + offset);
15. return 0;
16. }
18. set\_pull\_up(REG\_GPIO\_BASE，GPIO1\_IO01\_OFFSET,SET\_PULL\_UP);</span>

<span style="font-size:12px;">#define SET\_PULL\_UP 0x01

#define SET\_PULL\_DOWN 0x00

#define REG\_GPIO\_BASE 0x8e000000

#define GPIO1\_IO01\_OFFSET 0x400

int set\_pull\_up(unsigned int reg\_base，unsigned int offset,int up)

{

unsigned int gpio\_base;

unsigned int gpio;

gpio\_base = ioreamap(reg\_base,SIZE\_4K);//调用ioreamap映射GPIO空间到内存，映射大小根据实际需求而定

gpio = gpio\_base + offset;

gpio |= up; //将某个GPIO拉高，根据具体的寄存器操作而定

\_\_raw\_writel(gpio,gpio\_base + offset);

return 0;

}

set\_pull\_up(REG\_GPIO\_BASE，GPIO1\_IO01\_OFFSET,SET\_PULL\_UP);</span>

//拉低某个GPIO

1. #define SET\_PULL\_UP 0x01
2. #define SET\_PULL\_DOWN 0x00
3. #define REG\_GPIO\_BASE 0x8e000000
4. #define GPIO1\_IO01\_OFFSET 0x400
6. int set\_pull\_down(unsigned int reg\_base，unsigned int offset,int down)
7. {
8. unsigned int gpio\_base;
9. unsigned int gpio;
11. gpio\_base = ioreamap(reg\_base,SIZE\_4K);//调用ioreamap映射GPIO空间到内存，映射大小根据实际需求而定
12. gpio = gpio\_base + offset;
13. gpio &= down;      //将某个GPIO拉低，根据具体的寄存器操作而定
14. \_\_raw\_writel(gpio,gpio\_base + offset);
15. return 0;
16. }
18. set\_pull\_up(REG\_GPIO\_BASE，GPIO1\_IO01\_OFFSET,SET\_PULL\_DOWN);

#define SET\_PULL\_UP 0x01

#define SET\_PULL\_DOWN 0x00

#define REG\_GPIO\_BASE 0x8e000000

#define GPIO1\_IO01\_OFFSET 0x400

int set\_pull\_down(unsigned int reg\_base，unsigned int offset,int down)

{

unsigned int gpio\_base;

unsigned int gpio;

gpio\_base = ioreamap(reg\_base,SIZE\_4K);//调用ioreamap映射GPIO空间到内存，映射大小根据实际需求而定

gpio = gpio\_base + offset;

gpio &= down; //将某个GPIO拉低，根据具体的寄存器操作而定

\_\_raw\_writel(gpio,gpio\_base + offset);

return 0;

}

set\_pull\_up(REG\_GPIO\_BASE，GPIO1\_IO01\_OFFSET,SET\_PULL\_DOWN);