RICO-3288 GPIO

Edit by Jason Wang

GPIO

1. Introduction¶

GPIO is a programmable General Purpose Programming I/O peripheral. This component is an APB slave device. GPIO controls the output data and direction of external I/O pads. It also can read back the data on external pads using memory-mapped registers.

GPIO supports the following features:

- 32 bits APB bus width
- 32 independently configurable signals
- Separate data registers and data direction registers for each signal
- Software control for each signal, or for each bit of each signal
- Configurable interrupt mode

2. How to Use¶

Here will explain the GPIO pin definition and kernel GPIO API on the RIOC-3288 board. Figure 2.1 shows the names of GPIO from CPU side.

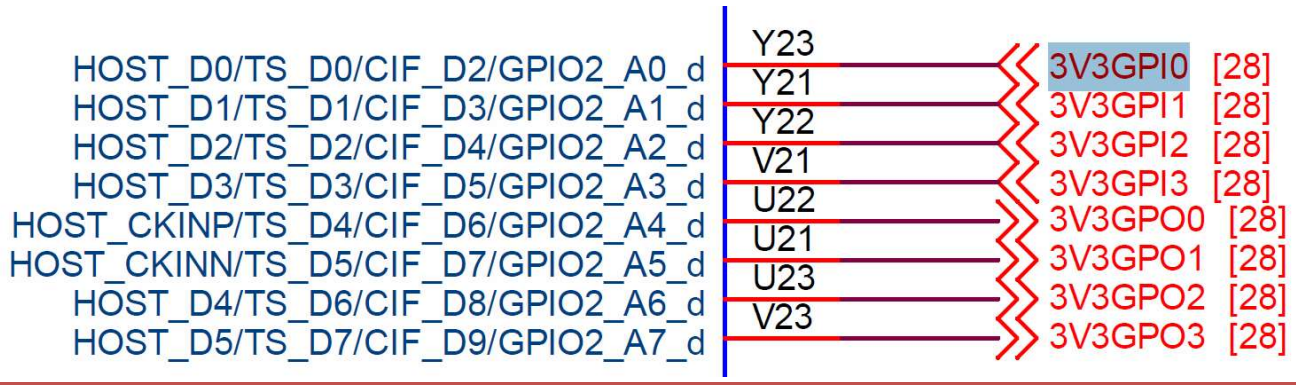

Figure 2.1

Here is a list of GPIO related functions in Kernel, as shown in the following table 2.2

FUNCTION NAME	Features
int gpio_request(unsigned gpio, const char *label);	Requesting a gpio number with kernel for sysfs, you need to set the gpio number and label.
int gpio_direction_input(unsigned gpio);	Set GPIO to Input mode
int gpio_direction_output(unsigned gpio, int value);	Set GPIO to Output mode
void gpio_set_value(unsigned gpio, int value);	Set the value of the GPIO output to high or low
int gpio_get_value(unsigned gpio);	Read the value of GPIO Input
int gpio_is_valid(int number);	Confirm whether the GPIO is repeatedly applied or occupied
int gpio_export(unsigned gpio, bool direction_may_change);	Create a sysfs link to an exported GPIO node

Table 2.2

Using GPIO2_A0 as an example, you must convert GPIO2_A0 to a normal GPIO number before use. A formula is provided here for easy conversion.

GPIO**X**_**Y**.**Z** = (32*__X__) -8 + (**Y**-1)*8 + **Z** ; (Y=A=1 ; Y=B=2; Y=C=3; Y=D=4)

So GPIO2_A0 is (32*2) -8 + (1-1)*8 +0 = 56 . Now you can start setting the input/output mode and value of GPIO-56 in Kernel.

GPIO-56 is set to high-level output such as table 2.3

```
if (!gpio_is_valid( 56 )) {
        printk("invalid gpio: %d\n", 56);
        return NULL;
    }
```

```
        ret = gpio_request( 56, "CN31_GPIO1");
                if (ret < 0) {
                        dev_err(RICO->dev,"Failed to request gpio %d with ret:""%d\n", 56, ret);
                }
                gpio_direction_output(56, 1);
                 gpio_set_value( 56,1 );
                gpio_export(56, 1);


        }
```

Table 2.3

GPIO-56 is set to input mode such as table 2.4

```
if (!gpio_is_valid( 56 )) {
                        printk("invalid gpio: %d\n", 56);
                        return NULL;
                }
ret = gpio_request( 56, "CN31_GPIO1");
                if (ret < 0) {
                        dev_err(RICO->dev,"Failed to request gpio %d with ret:""%d\n", 56, ret);
                }
                gpio_direction_intpu (56);
                 val = gpio_get_value( 56 );
                gpio_export(56, 1);


        }
```

Table 2.4