WF88E GPIO Developer Guide

GPIO Access

The following APIs can only be used in kernel space. The `outputgpio.c` and `inputgpio.c` files use the sysfs GPIO interface.

GPIO software mapping

GPIO	MCU internal ports	GPIO Num	Level state	Function	Describe	Node Type
GPIO0	PB29	61	Default pull-up	Power Enable Control	LR71 and BLE60-M power enable control, default kernel pull-up. This is a weak pull-up and cannot drive the power chip to enable it. That is to say, the LR71 and BLE60-M power supplies are disabled by default.	GateWay
GPIO1	PB30	62	Default pull-up	LINK	After the file system starts, light up the LED4 corresponding to that pin.	GateWay
GPIO2	PB28	60	Default pull-up	PHY Reset	Reset the Ethernet PHY chip.	GateWay Router
GPIO3	PB27	59	Default pull-up	Tx/Rx status	Blink on MQTT or TCP application layer data RX/TX activity. But this pin is connected to the LR71 and BLE60-M reset pins. The effect can be seen when the LED is connected to the Node, but it cannot be seen when the LED is not connected to the Gateway. However, it does not affect LR71 and BLE6-M.	Node GateWay Router

GPIO4	PB26	58	Default pull-up	Reset	After the system is started, press the Reset button for about 1 second and then lift it up to reset the system. If the Reset button is pressed and held down for approximately 10 seconds, the system will restore the factory set parameters.	GateWay
GPIO5	PB25	57	Default pull-up	STATE	LED status display: default off (This PIN output high level), lit after connecting to the mesh network, flashing after successful TCP/MQTT connection. The mode of "Station connects to AP", LED flashs.	Node Router
					LED5 status display: The LED lights up after the system is powered on and turns off after startup.Default off (This PIN output low level), lit after connecting to the mesh network, flashing after successful TCP/MQTT connection.	GateWay
GPIO6	PB17	49	Default pull-up			Pin undefined
GPIO7	PB18	50	Default pull-up			Pin undefined

Node: Mesh station, Station connects to AP

GateWay: Mesh Gateway, WF288LoRaNetGateway

1. Example 1: GPIO Output (LED Control)

1.1. outputgpio.c

```
/* outputgpio.c
 * This file is part of outputgpio
 * Copyright (c) 2026, Amp'edRF
* ============================================================
 * Read the file COPYING for details
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

#define LED_GPIO 50  // GPIO number used for LED control, GPIO7 of
WF88-EX

/**
 * @brief  Export GPIO pin to userspace via sysfs
 * @param  gpio: GPIO number to be exported
 * @return 0 on success, -1 on failure
 */
static int gpio_export(int gpio)
{
    int fd = open("/sys/class/gpio/export", O_WRONLY);
    if (fd < 0) {
        perror("export open failed");
        return -1;
    }

    char buf[4];
    snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, strlen(buf));
    close(fd);
```

```
    return 0;
}

/**
 * @brief  Set GPIO direction (input/output)
 * @param  gpio: GPIO number
 * @param  dir: "in" or "out" direction string
 * @return 0 on success, -1 on failure
 */
static int gpio_direction(int gpio, const char *dir)
{
    char path[64];
    snprintf(path, sizeof(path), "/sys/class/gpio/gpio%d/direction",
gpio);

    int fd = open(path, O_WRONLY);
    if (fd < 0) {
        perror("direction open failed");
        return -1;
    }

    write(fd, dir, strlen(dir));
    close(fd);

    return 0;
}

/**
 * @brief  Set GPIO output value (high/low)
 * @param  gpio: GPIO number
 * @param  value: 1 = high, 0 = low
 * @return 0 on success, -1 on failure
 */
static int gpio_write(int gpio, int value)
{
    char path[64];
    snprintf(path, sizeof(path), "/sys/class/gpio/gpio%d/value", gpio);
```

```
    int fd = open(path, O_WRONLY);
    if (fd < 0) {
        perror("value open failed");
        return -1;
    }

    char val = (value != 0) ? '1' : '0';
    write(fd, &val, 1);
    close(fd);

    return 0;
}

/**
 * @brief  Unexport GPIO pin from userspace
 * @param  gpio: GPIO number to be unexported
 * @return none
 */
static void gpio_unexport(int gpio)
{
    int fd = open("/sys/class/gpio/unexport", O_WRONLY);
    if (fd < 0) {
        perror("unexport open failed");
        return;
    }

    char buf[4];
    snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, strlen(buf));
    close(fd);
}

/**
 * @brief  Main function: LED control via userspace GPIO
 * @return 0 on normal exit
 */
int main()
```

```
{
    printf("LED Control Start\n");

    // Export GPIO to userspace
    gpio_export(LED_GPIO);
    sleep(1);

    // Set GPIO as output mode
    gpio_direction(LED_GPIO, "out");

    // Turn LED ON
    gpio_write(LED_GPIO, 1);
    printf("LED ON\n");
    sleep(3);

    // Turn LED OFF
    gpio_write(LED_GPIO, 0);
    printf("LED OFF\n");
    sleep(3);

    // Turn LED ON
    gpio_write(LED_GPIO, 1);
    printf("LED ON\n");
    sleep(3);

    // Release GPIO resource
    gpio_unexport(LED_GPIO);

    printf("LED Control End\n");
    return 0;
}
```

1.2. Compile

User:~/Ingenic/t23n/sdk/outputgpio$ make

mips-linux-gnu-gcc -Wall -g -O2 -O3 -fno-omit-frame-pointer -c -o outputgpio.o outputgpio.c

mips-linux-gnu-gcc -Wall -g -O2 -O3 -fno-omit-frame-pointer -o outputgpio outputgpio.o

1.3. View file

User:~/Ingenic/t23n/sdk/outputgpio$ ll

total 44

drwxrwxr-x 2 User 4096 4 月 9 15:53 ./

drwxrwxr-x 8 User 4096 4 月 10 18:12 ../

-rw-rw-r-- 1 User 749 4 月 9 14:43 Makefile

-rwxrwxr-x 1 User 12724 4 月 9 15:53 outputgpio*

-rw-rw-r-- 1 User 2893 4 月 9 17:45 outputgpio.c

-rw-rw-r-- 1 User 10780 4 月 9 15:53 outputgpio.o

User:~/Ingenic/t23n/sdk/outputgpio$

1.4. Create a file system

Copy the “outputgpio” to the “fs/system/system/bin/” directory and generate a file system.

User:~/Ingenic/t23n/sdk/outputgpio$ cp -a outputgpio ../fs/system/system/bin/

User:~/Ingenic/t23n/sdk/outputgpio$

Please refer to the "Creating File System" and "WF88-EX_USBClonerGuide" documents.

1.5. Run

[root@Ingenic-g1_1:~]# outputgpio

LED Control Start

LED ON

LED OFF

LED ON

LED Control End

[root@Ingenic-g1_1:~]#

```
#include <linux/gpio.h>

#define LED_GPIO 10

int led_init(void)
{
   int ret;
   ret = gpio_request(LED_GPIO, "my_led");
   if (ret < 0) {
      printk("gpio request failed\n");
      return ret;
   }

   gpio_direction_output(LED_GPIO, 0);
   gpio_set_value(LED_GPIO, 1);
   return 0;
}

void led_exit(void)
{
   gpio_set_value(LED_GPIO, 0);
   gpio_free(LED_GPIO);
}
```

2. Example 2: GPIO Input (Key Reading)

2.1. inputgpio.c

```
/* inputgpio.c
 * This file is part of inputgpio
 * Copyright (c) 2026, Amp'edRF
* ============================================================
 * Read the file COPYING for details
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
```

```
#define INPUT_GPIO 50  // GPIO number used for input read, GPIO7 of
WF88-EX

/**
 * @brief  Export GPIO pin to userspace via sysfs
 * @param  gpio: GPIO number to be exported
 * @return 0 on success, -1 on failure
 */
static int gpio_export(int gpio)
{
    int fd = open("/sys/class/gpio/export", O_WRONLY);
    if (fd < 0) {
        perror("export open failed");
        return -1;
    }

    char buf[4];
    snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, strlen(buf));
    close(fd);

    return 0;
}

/**
 * @brief  Set GPIO direction (input/output)
 * @param  gpio: GPIO number
 * @param  dir: "in" or "out" direction string
 * @return 0 on success, -1 on failure
 */
static int gpio_direction(int gpio, const char *dir)
{
    char path[64];
    snprintf(path, sizeof(path), "/sys/class/gpio/gpio%d/direction",
gpio);

    int fd = open(path, O_WRONLY);
    if (fd < 0) {
```

```
        perror("direction open failed");
        return -1;
    }

    write(fd, dir, strlen(dir));
    close(fd);

    return 0;
}

/**
 * @brief  Read GPIO input value (high/low)
 * @param  gpio: GPIO number
 * @return 0 = low, 1 = high, -1 on failure
 */
static int gpio_read(int gpio)
{
    char path[64];
    snprintf(path, sizeof(path), "/sys/class/gpio/gpio%d/value", gpio);

    int fd = open(path, O_RDONLY);
    if (fd < 0) {
        perror("value open failed");
        return -1;
    }

    char val;
    read(fd, &val, 1);
    close(fd);

    return (val == '1') ? 1 : 0;
}

/**
 * @brief  Unexport GPIO pin from userspace
 * @param  gpio: GPIO number to be unexported
 * @return none
 */
```

```
static void gpio_unexport(int gpio)
{
    int fd = open("/sys/class/gpio/unexport", O_WRONLY);
    if (fd < 0) {
        perror("unexport open failed");
        return;
    }

    char buf[4];
    snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, strlen(buf));
    close(fd);
}

/**
 * @brief  Main function: Read GPIO input state via userspace GPIO
 * @return 0 on normal exit
 */
int main()
{
    printf("GPIO Input Read Start\n");

    // Export GPIO to userspace
    gpio_export(INPUT_GPIO);
    sleep(1);

    // Set GPIO as input mode
    gpio_direction(INPUT_GPIO, "in");

    // Read GPIO state 5 times in loop
    for (int i = 0; i < 5; i++) {
        int value = gpio_read(INPUT_GPIO);
        if (value == 1) {
            printf("GPIO %d State: HIGH (1)\n", INPUT_GPIO);
        } else {
            printf("GPIO %d State: LOW (0)\n", INPUT_GPIO);
        }
        sleep(2); // Read every 2 seconds
```

```
    }

    // Release GPIO resource
    gpio_unexport(INPUT_GPIO);

    printf("GPIO Input Read End\n");
    return 0;
}
```

2.2. Compile

Carter@amped:~/Ingenic/t23n/sdk/inputgpio$ make

mips-linux-gnu-gcc -Wall -g -O2 -O3 -fno-omit-frame-pointer -c -o inputgpio.o inputgpio.c

mips-linux-gnu-gcc -Wall -g -O2 -O3 -fno-omit-frame-pointer -o inputgpio inputgpio.o

Carter@amped:~/Ingenic/t23n/sdk/inputgpio$

2.3. View file

Carter@amped:~/Ingenic/t23n/sdk/inputgpio$ ll

total 44

drwxrwxr-x 2 Carter Carter 4096 4 月 9 16:43 ./

drwxrwxr-x 8 Carter Carter 4096 4 月 10 18:12 ../

-rwxrwxr-x 1 Carter Carter 12840 4 月 9 16:43 inputgpio*

-rw-rw-r-- 1 Carter Carter 2937 4 月 9 17:45 inputgpio.c

-rw-rw-r-- 1 Carter Carter 10764 4 月 9 16:43 inputgpio.o

-rw-rw-r-- 1 Carter Carter 745 4 月 9 16:12 Makefile

Carter@amped:~/Ingenic/t23n/sdk/inputgpio$

2.4. Create a file system

Copy the “inputgpio” to the “fs/system/system/bin/” directory and generate a file system.

Carter@amped:~/Ingenic/t23n/sdk/inputgpio$ cp -a inputgpio ../fs/system/system/bin/

Carter@amped:~/Ingenic/t23n/sdk/inputgpio$

Please refer to the "Creating File System" and "WF88-EX_USBClonerGuide" documents.

2.5. Run

User can connect GPIO7 to high and low levels and execute commands for testing.

[root@Ingenic-g1_1:~]# inputgpio

GPIO Input Read Start

GPIO 50 State: HIGH (1)

GPIO 50 State: LOW (0)

GPIO 50 State: HIGH (1)

GPIO 50 State: LOW (0)

GPIO 50 State: LOW (0)

GPIO Input Read End

[root@Ingenic-g1_1:~]#

2.6. Meshconnect.sh

If users need to automatically run a compiled program, add it to the appropriate location in the meshconnect.sh script.

"meshconnect.sh" is in the "sdk/fs/system/system/etc/config" directory.