Device Driver_양영식

NXSOL-OJT-2024

Exported on 02/27/2024

Table of Contents

1 INDEX 3
2 0. UPDATE HISTORY 4
3 1. Introduction 5
4 2. Implemented features 6
4.1 2.1 Feature list 6
4.2 2.2 Command manual 6
4.3 2.3 Flow diagram 6
5 3. Target GPIO pins 8
6 4. Source 9
6.1 4.1 Source code 9
6.2 4.2 Add the source code 12
6.3 4.3 Existing APIs used in the source code 13
7 5. Test 14
7.1 5.1 Test scenario 14
7.2 5.2 Test code 15
7.3 5.3 Run test 19
7.4 5.4 Test result 19
8 Reference 23

1 INDEX

2 0. UPDATE HISTORY

EVENT	Date	Description	Author
create	2024.2.20	create	@NXKR_YoungSikYang
modify	2024.2.21	modified the contents according to the template	@NXKR_YoungSikYang
modify	2024.2.21	▪ corrected wrong indentation ▪ added source code ▪ added gpio-uclass.c reference	@NXKR_YoungSikYang
modify	2024.2.22	▪ modified implemented features ▪ deleted memory location ▪ modified test scenario	@NXKR_YoungSikYang
modify	2024.2.23	▪ Modified command manual so that it can be clearer ▪ Corrected a typo in flow diagram ▪ Added more explanation to each section	@NXKR_YoungSikYang

3 1. Introduction

- Final assignment of u-boot training.
- Board: BTC08(s5p6818)

Utilizing the GPIO device driver implemented in gpio-uclass.c, I have developed features as specified in the assignment requirements and tested them using u-boot commands.

4 2. Implemented features

The features are related to controlling or querying GPIOs.

4.1 2.1 Feature list

The features can be classified into 3 categories

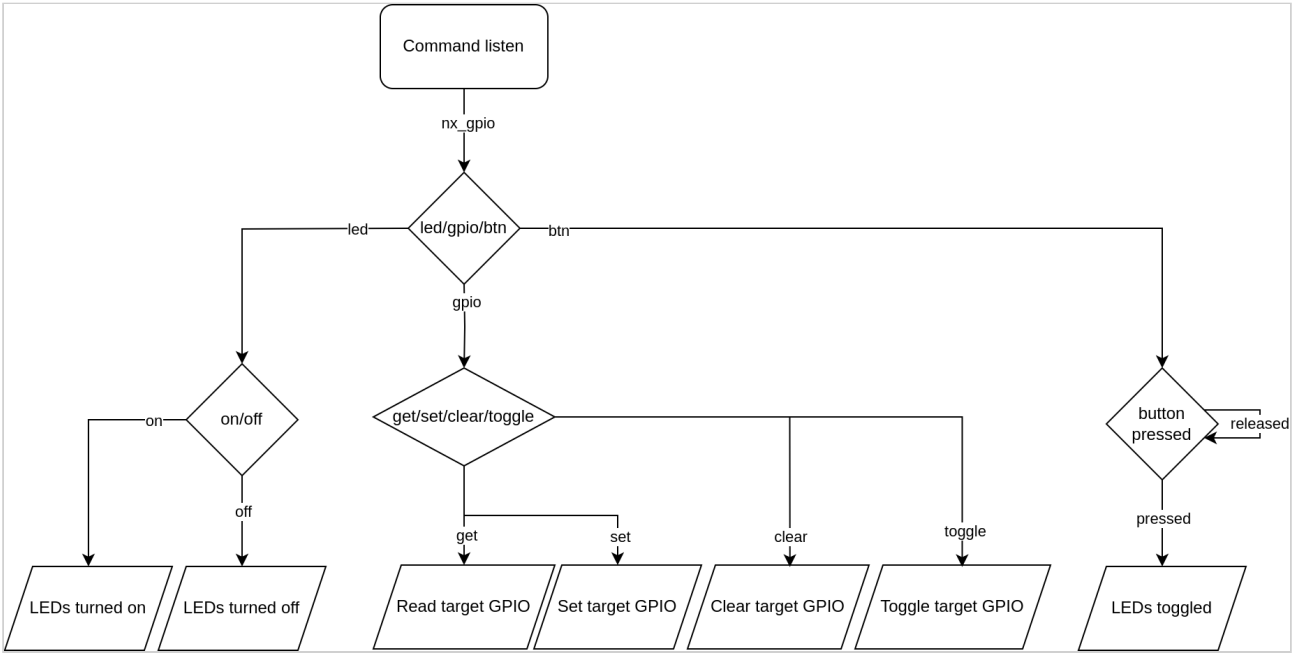
- GPIO input/output (GPIOA, GPIOB, GPIOC, GPIOD, GPIOE)
- LED control (AP_LED0, AP_LED1)
- Button detection (AP_KEY0)

4.2 2.2 Command manual

Command	**Action**
nx_gpio led <on|off>	Turn the LEDs on or off
nx_gpio <set|clear|toggle|get> <num>	set|clear|toggle|get <num> GPIO
nx_gpio btn	Wait for button presss and toggle the LEDs

4.3 2.3 Flow diagram

This diagram visualizes the behavior of each command.


Command listen
nx_gpio
led/gpio/btn
led
btn
gpio
on/off
on
off
get/set/clear/toggle
get
set
clear
toggle
button pressed
released
pressed
LEDs turned on
LEDs turned off
Read target GPIO
Set target GPIO
Clear target GPIO
Toggle target GPIO
LEDs toggled

5 3. Target GPIO pins

This picture is from Page 6 of tsb1101_control_v100_rel_20181126

6 4. Source

6.1 4.1 Source code

common/cmd_nx_gpio.c

```
#include <common.h>
#include <command.h>
#include <stdarg.h>
#include <errno.h>
#include <dm.h>
#include <asm/gpio.h>

#define BTN 3
#define LED0    4
#define LED1    5
#define ON  0
#define OFF 1

static int gpios_toggle(int gpio, ...)
{
    va_list args;
    va_start(args, gpio);
    int ret;
    while (gpio != -1) {
        if (gpio_request(gpio, "cmd_nx_gpio")) {
            printf("Failed to request gpio %i\n", gpio);
            return -1;
        }
        ret = gpio_get_value(gpio);
        if (ret < 0) {
            printf("Failed to get gpio %i value\n", gpio);
            gpio_free(gpio);
            return ret;
        }
        ret = gpio_set_value(gpio, !ret);
        if (ret < 0) {
            printf("Failed to toggle gpio %i value\n", gpio);
            gpio_free(gpio);
            return ret;
        }
        printf("gpio %i value is %d\n", gpio, !ret);
        gpio_free(gpio);
        gpio = va_arg(args, int);
```

```
    }
    va_end(args);
    return ret;
}

static int gpios_set_value(int gpio_value, int gpio, ...)
{
    va_list args;
    va_start(args, gpio);
    int ret;
    while (gpio != -1) {
        if (gpio_request(gpio, "cmd_nx_gpio")) {
            printf("Failed to request gpio %i\n", gpio);
            return -1;
        }
        ret = gpio_set_value(gpio, gpio_value);
        if (ret < 0) {
            printf("Failed to set gpio %i value\n", gpio);
            gpio_free(gpio);
            return ret;
        }
        printf("gpio %i value is %d\n", gpio, gpio_value);
        gpio_free(gpio);
        gpio = va_arg(args, int);
    }
    va_end(args);
    return ret;
}

int do_nx_gpio(cmd_tbl_t *cmdtp, int flag, int argc, char * const argv[])
{
    int gpio_value, gpio, ret;
    /* Not enough args */
    if (argc < 2) {
        printf("Usage: %s\n", cmdtp->usage);
        return CMD_RET_FAILURE;
    }

    if (strcmp(argv[1], "led") == 0) {
        if (argc < 3) {
            printf("Usage: %s\n", cmdtp->usage);
            return CMD_RET_FAILURE;
        }
        if (strcmp(argv[2], "on") == 0) {
            if (gpios_set_value(ON, LED0, LED1, -1))
                return CMD_RET_FAILURE;
        } else if (strcmp(argv[2], "off") == 0) {
            if (gpios_set_value(OFF, LED0, LED1, -1))
                return CMD_RET_FAILURE;
        } else {
            printf("Usage: %s\n", cmdtp->usage);
            return CMD_RET_FAILURE;
```

```
    }
} else if (strcmp(argv[1], "gpio") == 0) {
    if (argc < 4) {
        printf("Usage: %s\n", cmdtp->usage);
        return CMD_RET_FAILURE;
    }
    ret = gpio_lookup_name(argv[3], NULL, NULL, &gpio);
    if (gpio < 0) {
        printf("Usage: %s\n", cmdtp->usage);
        return CMD_RET_FAILURE;
    }
    if (strcmp(argv[2], "get") == 0) {
        if (gpio_request(gpio, "cmd_nx_gpio")) {
            printf("Failed to request gpio %i\n", gpio);
            return CMD_RET_FAILURE;
        }
        gpio_value = gpio_get_value(gpio);
        if (gpio_value < 0) {
            printf("Failed to get gpio %i value\n", gpio);
            gpio_free(gpio);
            return CMD_RET_FAILURE;
        }
        printf("gpio %i value is %d\n", gpio, gpio_value);
        gpio_free(gpio);
    } else if (strcmp(argv[2], "set") == 0) {
        if (gpios_set_value(1, gpio, -1))
            return CMD_RET_FAILURE;
    } else if (strcmp(argv[2], "clear") == 0) {
        if (gpios_set_value(0, gpio, -1))
            return CMD_RET_FAILURE;
    } else if (strcmp(argv[2], "toggle") == 0) {
        if (gpios_toggle(gpio, -1))
            return CMD_RET_FAILURE;
    } else {
        printf("Usage: %s\n", cmdtp->usage);
        return CMD_RET_FAILURE;
    }
} else if (strcmp(argv[1], "btn") == 0) {
    printf("Waiting for button press...\n");
    if (gpio_request(BTN, "cmd_nx_gpio")) {
        printf("Failed to request gpio %i\n", gpio);
        return CMD_RET_FAILURE;
    }
    /* Polling */
    while (1) {
        gpio_value = gpio_get_value(BTN);
        if (gpio_value < 0) {
            printf("Failed to get gpio %i value\n", 3);
            gpio_free(BTN);
            return CMD_RET_FAILURE;
        }
        if (gpio_value == 0) {
```

```
                gpio_free(BTN);
                break;
            }
        }
        /* Toggle LEDs */
        if (gpios_toggle(LED0, LED1, -1))
            return CMD_RET_FAILURE;
    } else {
        printf("Usage: %s\n", cmdtp->usage);
        return CMD_RET_FAILURE;
    }
    return CMD_RET_SUCCESS;
}

U_BOOT_CMD(nx_gpio, 4, 0, do_nx_gpio,
    "Query and control gpio pins",
    "\n- led on|off: Turn the LEDs on|off\n"
    "- gpio set|clear|toggle|get <num>: Set|Clear|Toggle|Get <num> gpio\n"
    "- btn: Wait for button press and toggle 2 LEDs\n"
);
```

6.2 4.2 Add the source code

Configure so that the new source code can be compiled and included in u-boot through the build process.

`common/Makefile`

This adds the source to the compile list.

```
# My code
obj-$(CONFIG_CMD_NX_GPIO) += cmd_nx_gpio.o
obj-$(CONFIG_TEST_CMD_NX_GPIO) += test/cmd_nx_gpio.o
```

`common/kconfig`

This includes the source in u-boot.

```
config CMD_NX_GPIO
        bool "nx_gpio"
        default y
        help
                Turn on custom GPIO commands
config TEST_CMD_NX_GPIO
        bool "Test nx_gpio commands"
        default y
        help
                Test for custom GPIO commands
```

And then build

6.3 4.3 Existing APIs used in the source code

The source code above takes advantage of this API source

https://github.com/NexellCorp/u-boot-2016.01/blob/artik/drivers/gpio/gpio-uclass.c

7 5. Test

Each command is tested by visually checking the color of the LEDs and verifying the input of the target GPIO through the GPIO driver.

7.1 5.1 Test scenario

Case	Command	Expectation	Goal
get GPIOA3	nx_gpio gpio get 3	GPIOA3 is high with button released → button press -> GPIOA3 is low	GPIO input works
led off	nx_gpio led off	LEDs are on → LEDs are turned off	LED on/off works
led on	nx_gpio led on	LEDs are off → LEDs are turned on	LED on/off works
led abc	nx_gpio led abc	Command does not work (invalid command)	Invalid command does not work
set GPIOA4,5	nx_gpio gpio set 4 nx_gpio gpio set 5	GPIOA4,5 are 0 → GPIOA4,5 are set to 1(LEDs are turned off)	GPIO output works
clear GPIOA4,5	nx_gpio gpio clear 4 nx_gpio gpio clear 5	GPIOA4,5 are 1 → GPIOA4,5 are cleared to 0(LEDs are turned on)	GPIO output works
toggle GPIOA4,5	nx_gpio gpio toggle 4 nx_gpio gpio toggle 5	GPIOA4,5 are 0 → GPIOA4,5 are toggled to 1(LEDs are turned off)	GPIO output works
toggle GPIOA4,5 on button press	nx_gpio btn	GPIOA4,5 are 1 → button press -> GPIOA4,5 are toggled to 0(LEDs are turned on)	GPIO output works

7.2 5.2 Test code

test/cmd_nx_gpio.c

```c
#include <common.h>
#include <command.h>
#include <errno.h>
#include <dm.h>
#include <asm/gpio.h>

static void gpio_get(unsigned int *total_tests, unsigned int *passed_tests)
{
    printf("Test: gpio get\n");
    *total_tests += 1;
    /* Given: button connected to GPIO 3 is released */
    /* When: check the value of GPIO 3 */
    gpio_request(3, "cmd_nx_gpio");
    char gpio_value = gpio_get_value(3);
    /* Then: GPIO 3 is high */
    if (gpio_value == 1)
        printf("GPIO 3 is high.(released)\n");
    else {
        printf("Test failed: GPIO 3 is not high.\n");
        return;
    }
    printf("Waiting for button press...\n");
    /* Given: button connected to GPIO 3 is released */
    /* When: button is pressed */
    while (1) {
        gpio_value = gpio_get_value(3);
        if (gpio_value == 0)
            break;
    }
    /* Then: GPIO 3 is low */
    if (gpio_get_value(3) == 0) {
        printf("Test passed: GPIO 3 is low.(pressed)\n");
        ++*passed_tests;
    } else
        printf("Test failed: GPIO 3 is not low.\n");
    gpio_free(3);
}

static void led_off(unsigned int *total_tests, unsigned int *passed_tests)
{
    printf("Test: led off\n");
    *total_tests += 1;
    /* Given: LEDs are on */
```

```
    /* When: turn off the LEDs */
    run_command("nx_gpio led off", 0);
    /* Then: GPIO 4 and 5 are high */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    if (gpio_get_value(4) == 1 && gpio_get_value(5) == 1) {
        printf("Test passed: LEDs off.\n");
        ++*passed_tests;
    } else
        printf("Test failed: LEDs are not off.\n");
    gpio_free(4);
    gpio_free(5);
}

static void led_on(unsigned int *total_tests, unsigned int *passed_tests)
{
    printf("Test: led on\n");
    *total_tests += 1;
    /* Given: LEDs are off */
    /* When: turn on the LEDs */
    run_command("nx_gpio led on", 0);
    /* Then: GPIO 4 and 5 are low */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    if (gpio_get_value(4) == 0 && gpio_get_value(5) == 0) {
        printf("Test passed: LEDs on.\n");
        ++*passed_tests;
    } else
        printf("Test failed: LEDs are not on.\n");
    gpio_free(4);
    gpio_free(5);
}

static void led_abc(unsigned int *total_tests, unsigned int *passed_tests)
{
    int ret;
    printf("Test: led abc\n");
    *total_tests += 1;
    /* Given: LEDs are off */
    /* When: turn on the LEDs */
    ret = run_command("nx_gpio led abc", 0);
    /* Then: Invalid command */
    if (ret == 1) {
        printf("Test passed: Invalid command.\n");
        ++*passed_tests;
    } else if (ret == 0)
        printf("Test failed: Command not invalid.\n");
}

static void gpio_set(unsigned int *total_tests, unsigned int *passed_tests)
{
    printf("Test: gpio set\n");
```

```
    *total_tests += 1;
    /* Given: LEDs are on */
    /* When: setting GPIO 4 and 5 */
    run_command("nx_gpio gpio set 4", 0);
    run_command("nx_gpio gpio set 5", 0);
    /* Then: LEDs are off */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    if (gpio_get_value(4) == 1 && gpio_get_value(5) == 1) {
        printf("Test passed: GPIO 4 and 5 set.\n");
        ++*passed_tests;
    } else
        printf("Test failed: GPIO 4 and 5 are not set.\n");
    gpio_free(4);
    gpio_free(5);
}

static void gpio_clear(unsigned int *total_tests, unsigned int *passed_tests)
{
    printf("Test: gpio clear\n");
    *total_tests += 1;
    /* Given: LEDs are off */
    /* When: clear GPIO 4 and 5 */
    run_command("nx_gpio gpio clear 4", 0);
    run_command("nx_gpio gpio clear 5", 0);
    /* Then: LEDs are on */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    if (gpio_get_value(4) == 0 && gpio_get_value(5) == 0) {
        printf("Test passed: GPIO 4 and 5 cleared.\n");
        ++*passed_tests;
    } else
        printf("Test failed: GPIO 4 and 5 are not cleared.\n");
    gpio_free(4);
    gpio_free(5);
}

static void gpio_toggle(unsigned int *total_tests, unsigned int *passed_tests)
{
    printf("Test: gpio toggle\n");
    *total_tests += 1;
    /* Given: initial state of GPIO 4 and 5 */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    char gpio4_value = gpio_get_value(4);
    char gpio5_value = gpio_get_value(5);
    gpio_free(4);
    gpio_free(5);
    /* When: toggle GPIO 4 and 5 */
    run_command("nx_gpio gpio toggle 4", 0);
    run_command("nx_gpio gpio toggle 5", 0);
    gpio_free(4);
```

```
    gpio_free(5);
    /* Then: GPIO 4 and 5 are toggled */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    if (gpio4_value != gpio_get_value(4)
    && gpio5_value != gpio_get_value(5)) {
        printf("Test passed: GPIO 4 and 5 toggled.\n");
        ++*passed_tests;
    } else
        printf("Test failed: GPIO 4 and 5 are not toggled.\n");
    gpio_free(4);
    gpio_free(5);
}

static void btn(unsigned int *total_tests, unsigned int *passed_tests)
{
    printf("Test: btn\n");
    *total_tests += 1;
    /* Given: initial state of GPIO 4 and 5 /
    the button connected to GPIO 3 is released */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    char gpio4_value = gpio_get_value(4);
    char gpio5_value = gpio_get_value(5);
    gpio_free(4);
    gpio_free(5);
    /* When: wait for button to be pressed */
    run_command("nx_gpio btn", 0);
    /* Then: LEDs are toggled */
    gpio_request(4, "cmd_nx_gpio");
    gpio_request(5, "cmd_nx_gpio");
    if (gpio4_value != gpio_get_value(4) &&
    gpio5_value != gpio_get_value(5)) {
        printf("Button pressed and LEDs toggled.\n");
        ++*passed_tests;
    } else
        printf("Button press not recognized or LEDs not toggled.\n");
    gpio_free(4);
    gpio_free(5);
}

static void run_test(void (*test)(unsigned int *, unsigned int *),
unsigned int *total_tests, unsigned int *passed_tests)
{
    test(total_tests, passed_tests);
    printf("Press any key for the next test...\n--------------\n");
    while (!tstc())
        ;
    getc();
}

int do_test_nx_gpio(void)
```

```
{
    unsigned int total_tests = 0;
    unsigned int passed_tests = 0;
    run_test(gpio_get, &total_tests, &passed_tests);
    if (passed_tests == 0) /* Other tests depend on this one */
        return;
    run_test(led_off, &total_tests, &passed_tests);
    run_test(led_on, &total_tests, &passed_tests);
    run_test(led_abc, &total_tests, &passed_tests);
    run_test(gpio_set, &total_tests, &passed_tests);
    run_test(gpio_clear, &total_tests, &passed_tests);
    run_test(gpio_toggle, &total_tests, &passed_tests);
    btn(&total_tests, &passed_tests);
    printf("--------------\n"
    "Test completed. %u out of %u tests passed.\n",
    passed_tests, total_tests);
    return CMD_RET_SUCCESS;
}

U_BOOT_CMD(test_nx_gpio, 1, 0, do_test_nx_gpio,
    "Test cmd_nx_gpio",
    "\nRun tests for cmd_nx_gpio\n"
    ", which include gpio get, led on|off, gpio set|clear|toggle, and btn\n"
);
```

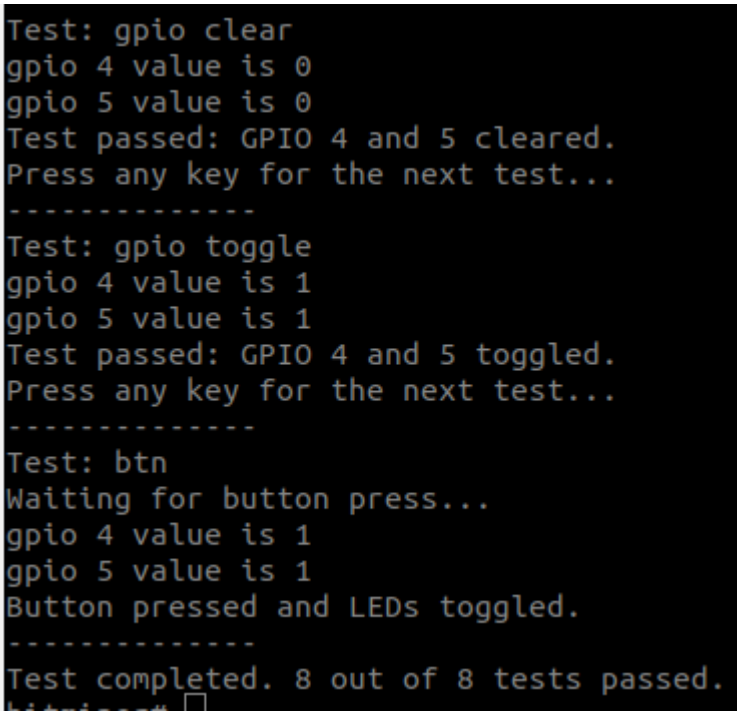
7.3 5.3 Run test

```
$ test_nx_gpio
```

7.4 5.4 Test result

The result table is below.

```
bitminer# test_nx_gpio
Test: gpio get
GPIO 3 is high.(released)
Waiting for button press...
Test passed: GPIO 3 is low.(pressed)
Press any key for the next test...
--------------
Test: led off
gpio 4 value is 1
gpio 5 value is 1
Test passed: LEDs off.
Press any key for the next test...
--------------
Test: led on
gpio 4 value is 0
gpio 5 value is 0
Test passed: LEDs on.
Press any key for the next test...
--------------
Test: led abc
Usage: Query and control gpio pins
Test passed: Invalid command.
Press any key for the next test...
--------------
Test: gpio set
gpio 4 value is 1
gpio 5 value is 1
Test passed: GPIO 4 and 5 set.
Press any key for the next test...
```

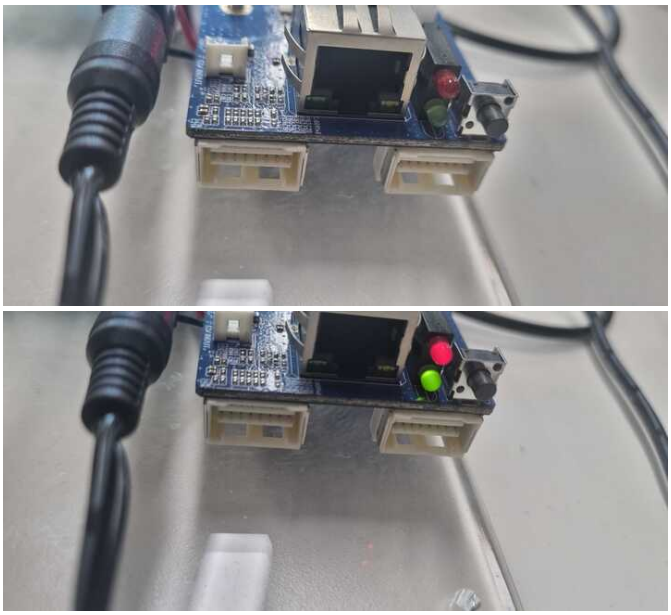


Test result table

- Pass: Test case worked as "expectation" of the test scenario table
- Fail: Test case did not work as "expectation" of the test scenario table

Case	Result
get GPIOA3 input	Pass
led off	Pass
led on	Pass
led abc	Pass
set GPIOA4,5	Pass
clear GPIOA4,5	Pass
toggle GPIOA4,5	Pass
toggle GPIOA4,5 on button press	Pass

LEDs successfully turned on and off before and after each GPIO output command



8 Reference

- tsb1101_control_v100_rel_20181126
- S5P6818_Datasheet_Ver142A
- https://docs.u-boot.org/en/latest/
- u-boot-2016.01