Device Driver\_양영식

NXSOL-OJT-2024

Exported on 2024-02-23 13:40:14

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 7

6 4. Source 8

6.1 4.1 Source code 8

6.2 4.2 Add the source code 11

6.3 4.3 Existing APIs used in the source code 11

7 5. Test 12

7.1 5.1 Test scenario 12

7.2 5.2  Test code 13

7.3 5.3 Run test 17

7.4 5.4 Test result 17

8 Reference 20

# INDEX

* [INDEX](#scroll-bookmark-3)
* [0. UPDATE HISTORY](#scroll-bookmark-4)
* [1. Introduction](#scroll-bookmark-5)
* [2. Implemented features](#scroll-bookmark-6)
  + [2.1 Feature list](#scroll-bookmark-7)
  + [2.2 Command manual](#scroll-bookmark-8)
  + [2.3 Flow diagram](#scroll-bookmark-9)
* [3. Target GPIO pins](#scroll-bookmark-10)
* [4. Source](#scroll-bookmark-11)
  + [4.1 Source code](#scroll-bookmark-12)
  + [4.2 Add the source code](#scroll-bookmark-13)
  + [4.3 Existing APIs used in the source code](#scroll-bookmark-14)
* [5. Test](#scroll-bookmark-15)
  + [5.1 Test scenario](#scroll-bookmark-16)
  + [5.2 Test code](#scroll-bookmark-17)
  + [5.3 Run test](#scroll-bookmark-18)
  + [5.4 Test result](#scroll-bookmark-19)
* [Reference](#scroll-bookmark-20)

# 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 |

# 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.

# 2. Implemented features

The features are related to controlling or querying GPIOs.

## 2.1 Feature list

The features can be classified into 3 categories

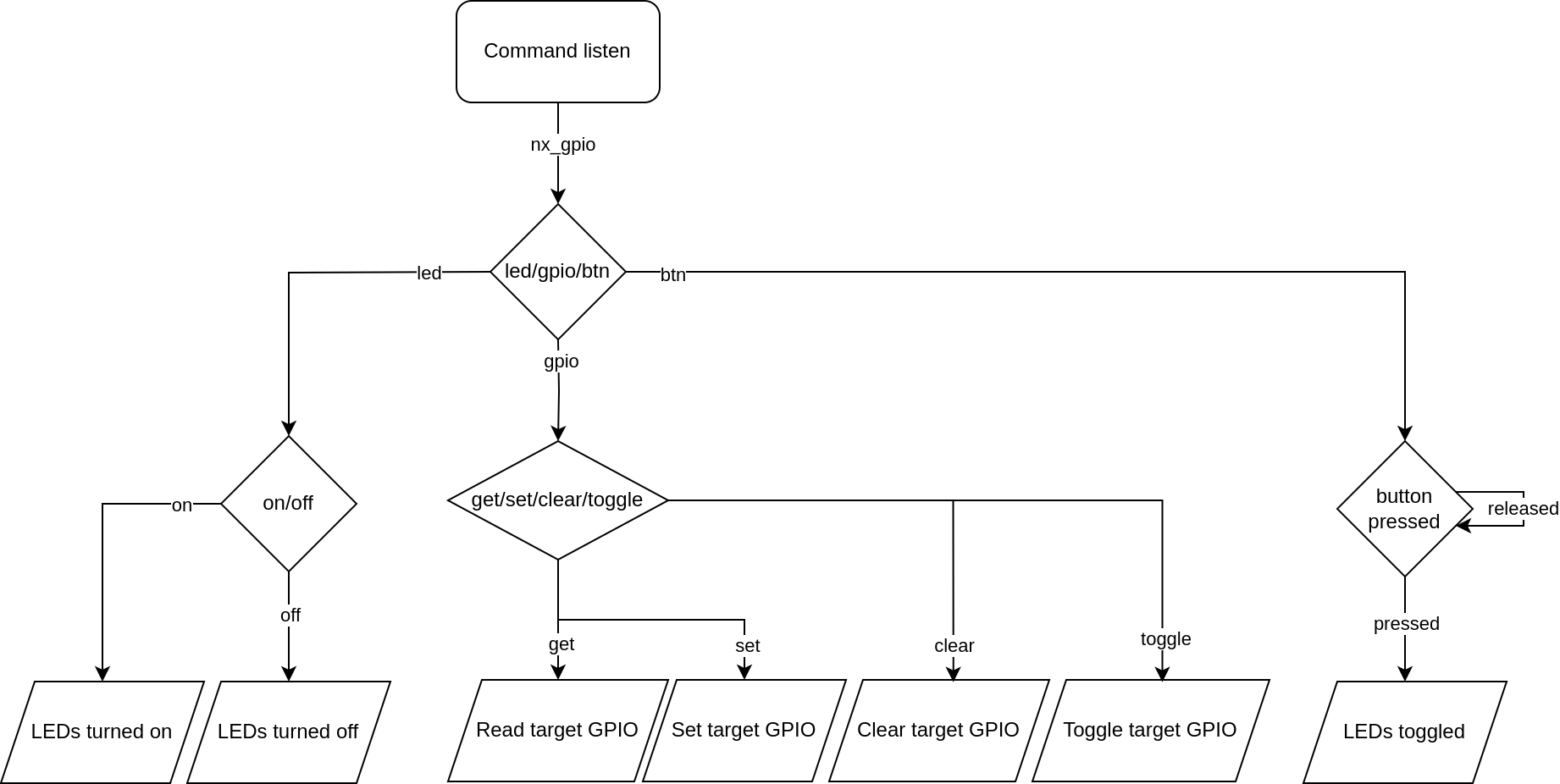
* GPIO input/output (GPIOA, GPIOB, GPIOC, GPIOD, GPIOE)
* LED control (AP\_LED0, AP\_LED1)
* Button detection (AP\_KEY0)

## 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 |

## 2.3 Flow diagram

This diagram visualizes the behavior of each command.



# 3. Target GPIO pins

This picture is from Page 6 of tsb1101\_control\_v100\_rel\_20181126.pdf





# 4. Source

## 4.1 Source code

|  |
| --- |
| #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"  ); |

Code Block 1 common/cmd\_nx\_gpio.c

## 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.

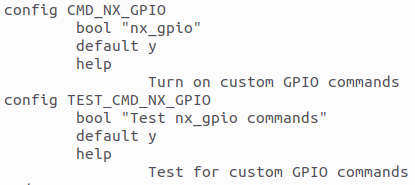
common/Makefile

This adds the source to the compile list.



common/kconfig

This includes the source in u-boot.



And then build

## 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

# 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.

## 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 |

## 5.2  Test code

|  |
| --- |
| #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"  ); |

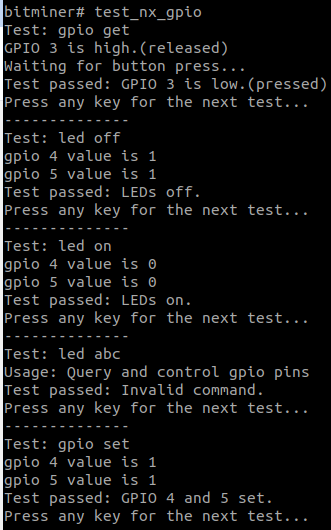
Code Block 2 test/cmd\_nx\_gpio.c

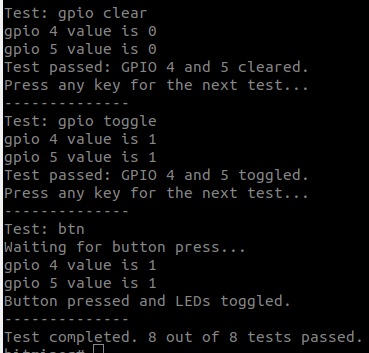
## 5.3 Run test

|  |
| --- |
| $ test\_nx\_gpio |

## 5.4 Test result

The result table is below.





**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**



# Reference

* tsb1101\_control\_v100\_rel\_20181126.pdf
* S5P6818\_Datasheet\_Ver142A.pdf
* https://docs.u-boot.org/en/latest/
* https://github.com/NexellCorp/u-boot-2016.01