

### **3. GPIO**

NXKR\_YoungSikYang

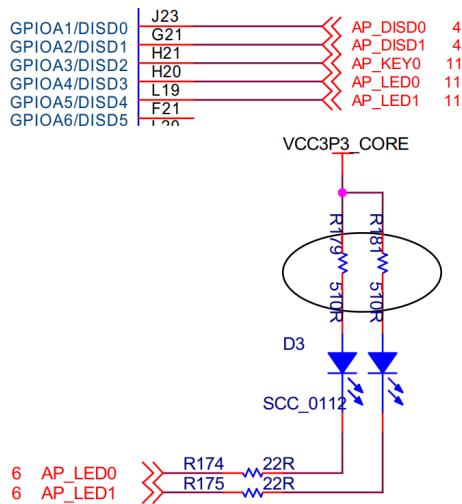
Exported on 02/27/2024

# Table of Contents

1	Check which GPIO the LED is connected to.....	3
2	Device tree.....	4
3	Data sheet .....	5
3.1	Check which memory address the GPIO is connected to .....	5
4	GPIO control .....	6
4.1	Command control.....	6
4.1.1	Method 1 .....	6
4.1.1.1	Check if defconfig has this line.....	6
4.1.1.2	Toggle the GPIO the LED is connected to .....	6
4.1.2	Method 2 .....	6
4.1.2.1	Turn off the LEDs by writing to the memory location.....	6
4.1.3	Before .....	6
4.1.4	After .....	7
4.2	Direct control in the u-boot source .....	7
4.2.1	Trace back the found function.....	8
4.2.2	Add the feature .....	8

# 1 Check which GPIO the LED is connected to

GPIOA4 <> AP\_LED0



## 2 Device tree

```
gpio_a:gpio@c001a000 {  
    compatible = "nexell,nexell-gpio";  
    reg = <0xc001a000 0x00000010>;  
    altr,gpio-bank-width = <32>;  
    gpio-bank-name = "gpio_a";  
    gpio-controller;  
    #gpio-cells = <2>;  
};
```

## 3 Data sheet

### 3.1 Check which memory address the GPIO is connected to

GPIO\_A starts from 0xc001a000

#### 16.5.1.1 GPIOxOUT

- Base Address: C001\_A000h (GPIOA)
- Base Address: C001\_B000h (GPIOB)
- Base Address: C001\_C000h (GPIOC)
- Base Address: C001\_D000h (GPIOD)
- Base Address: C001\_E000h (GPIOE)
- Address = Base Address + A000h, B000h, C000h, D000h, E000h, Reset Value = 0x0000\_0000

Name	Bit	Type	Description	Reset Value
GPIOXOUT	[31:0]	RW	GPIOx[31:0]: Specifies the output value in GPIOx output mode.  This bit should be set as "1" (Input mode) or "0" (Output mode) to use the Open drain pins in Input/Output mode. 0 = Low Level 1 = High Level	32'h0

## 4 GPIO control

### 4.1 Command control

#### 4.1.1 Method 1

##### 4.1.1.1 Check if defconfig has this line

```
| CONFIG_CMD_GPIO=y
```

##### 4.1.1.2 Toggle the GPIO the LED is connected to

```
gpio toggle 4
```

#### 4.1.2 Method 2

##### 4.1.2.1 Turn off the LEDs by writing to the memory location

```
md 0xc001a000 # Print  
mw 0xc001a000 30 # Write 0x30 = 0b00110000  
md 0xc001a000 # Print
```

#### 4.1.3 Before

```
c001a000: 00000000 01000030 00000000 00000000
```



#### 4.1.4 After

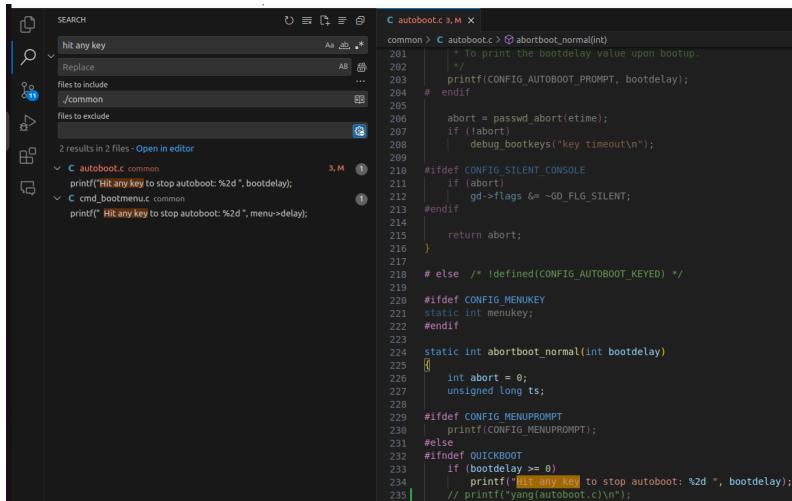
```
c001a000: 00000030 01000030 00000000 00000000
```



## 4.2 Direct control in the u-boot source

Find what function prints this to find the initial booting code

```
Hit any key to stop autoboot:
```



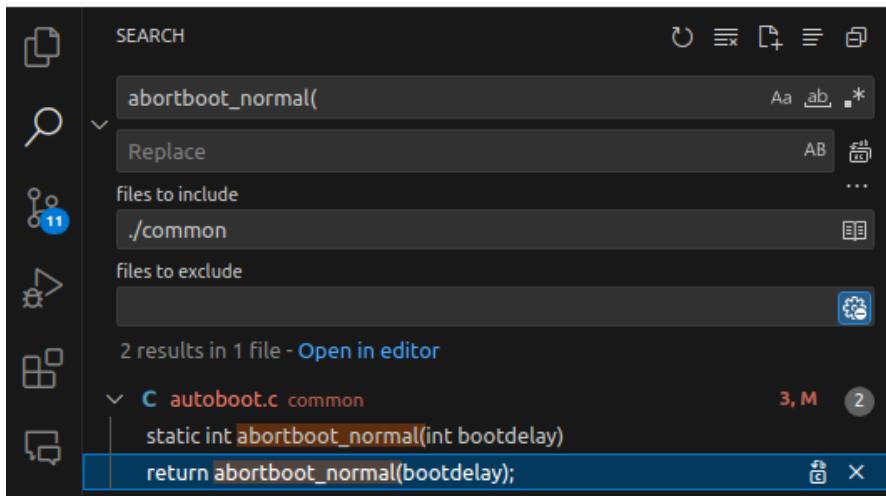
```

SEARCH hit any key
Replace AB ⌘U
Files to include ./common
Files to exclude
2 results in 2 files - Open in editor
C autoboot.c common
print("Hit any key to stop autoboot: %d", bootdelay);
C cmd_bootmenu.c common
printf(" Hit any key to stop autoboot: %d ", menu->delay);

common > C autoboot.c abortboot_normal(int)
201     * To print the bootdelay value upon bootup.
202     */
203     printf(CONFIG_AUTOBOOT_PROMPT, bootdelay);
204 #endif
205
206     abort = passwd_abort(etime);
207     if (!abort)
208         | debug_bootkeys("key timeout\n");
209
210 #ifdef CONFIG_SILENT_CONSOLE
211     if (abort)
212         gd->flags &= ~GD_FLG_SILENT;
213 #endif
214
215     return abort;
216 }
217
218 #else /* !defined(CONFIG_AUTOBOOT_KEYED) */
219
220 #ifdef CONFIG_MENUKEY
221     static int menukey;
222 #endif
223
224     static int abortboot_normal(int bootdelay)
225 {
226     int abort = 0;
227     unsigned long ts;
228
229 #ifdef CONFIG_MENUPROMPT
230     printf(CONFIG_MENUPROMPT);
231 #else
232 #ifndef QUICKBOOT
233     if (bootdelay >= 8)
234         printf(" Hit any key to stop autoboot: %d ", bootdelay);
235 // printf("yang(autoboot.c)\n");

```

#### 4.2.1 Trace back the found function



```

SEARCH abortboot_normal()
Replace AB ⌘U
Files to include ./common
Files to exclude
2 results in 1 file - Open in editor
C autoboot.c common
static int abortboot_normal(int bootdelay)
    return abortboot_normal(bootdelay);

```

#### 4.2.2 Add the feature

Run this function below `autoboot_command()` in `common/main.c`

```

void blink_LED() {
    int i=0, gpio_value=1;
    for(i=0; i<10; i++){
        printf("hello \n");
        // # Use the device driver functions of the GPIO
        // gpio_request(4, "cmd_gpio");
        // gpio_request(5, "cmd_gpio");
        // gpio_direction_output(4, gpio_value);
        // gpio_direction_output(5, gpio_value);
        // gpio_value=!gpio_value;
}

```

```
// # Bit control
*(char*)0xc001a000 ^= 1 << 4;
*(char*)0xc001a000 ^= 1 << 5;
mdelay(100);
}
```