3. GPIO

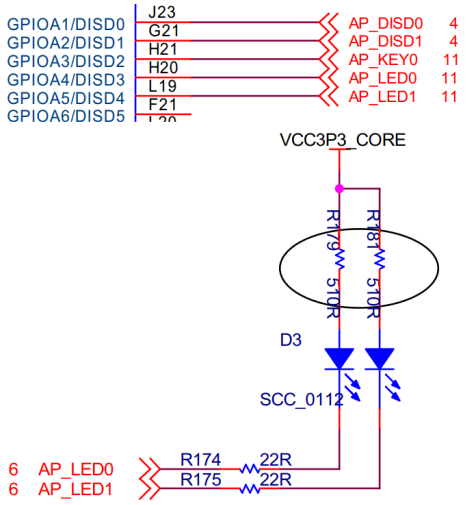
NXKR_YoungSikYang

Exported on 02/27/2024

Table of Contents

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
```

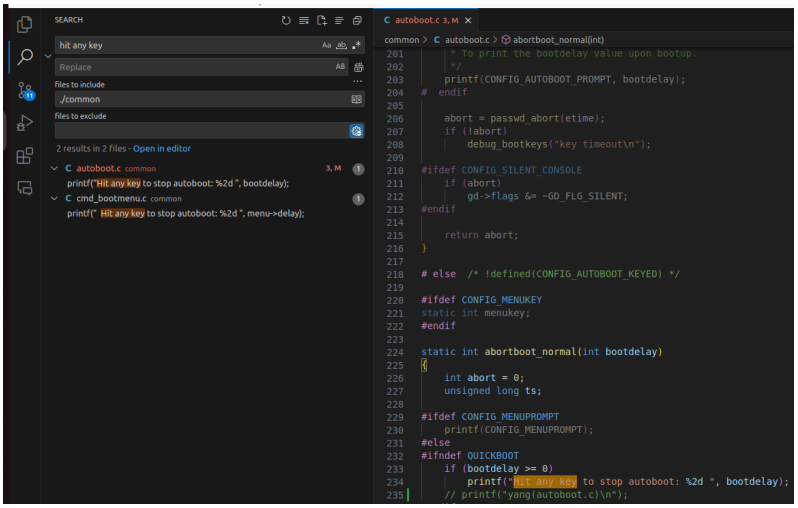
4.1.4 After

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

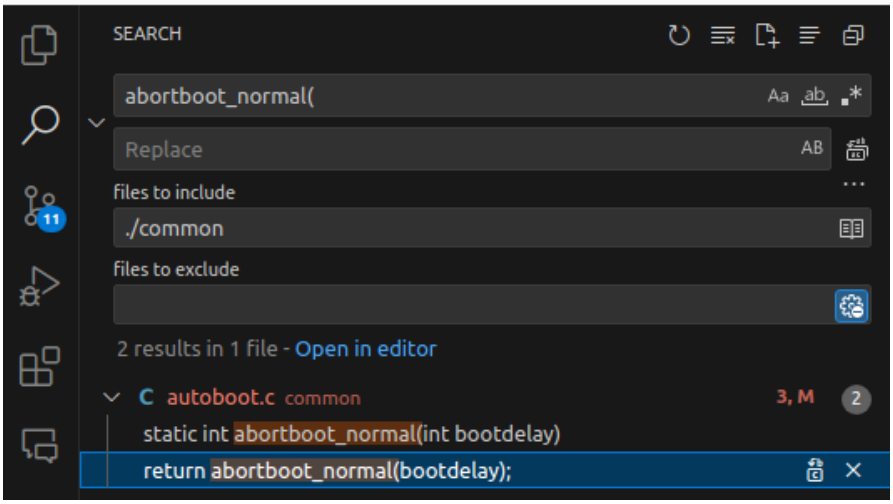
4.2 Direct control in the u-boot source

Find what function prints this to find the initial booting code

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



4.2.1 Trace back the found function



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);
    }
}
```