

# An Introduction to GPIO Programming

NUC980/NUC970 with Linux 5.10

江天文 9/24/2024

Joy of innovation  
**nuvoTon**

# | Configuring the Linux

- To access GPIO from user space, configure Linux to enable sysfs interface and support character device.
- Configure Linux within Buildroot  
**\$ make linux-menuconfig**

```
config - Linux/arm 5.10.140 Kernel Configuration
> Device Drivers > GPIO Support

GPIO Support
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus
----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M>
modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable

--- GPIO Support
(512) Maximum number of GPIOs for fast path
[ ] Debug GPIO calls
[*] /sys/class/gpio/... (sysfs interface)
[*] Character device (/dev/gpiochipN) support
[*] Support GPIO ABI Version 1
Memory mapped GPIO drivers --->
I2C GPIO expanders --->
MFD GPIO expanders --->
SPI GPIO expanders --->
USB GPIO expanders ----
< > GPIO Aggregator
< > GPIO Testing Driver

<Select> <Exit > <Help > <Save > <Load >
```

# Modifying the Device Tree

- Modify the Device Tree Source file  
<linux/arch/arm/boot/dts/nuc980.dtsi>
- In the device node entry  
apb:gpio:pinctrl\_gpio:gpio-eint,  
configure 3 custom GPIO  
pins: PB8, PB13 and PG15

```
1 0x08 0x0 0 /* PB8 */
1 0x0D 0x0 0 /* PB13 */
6 0x0F 0x0 0 /* PG15 */
```

- The GPIO pin number is  
defined in file  
<linux/arch/arm/mach-nuc980/include/mach/gpio.h>

```
gpio {
    pinctrl_gpio: gpio-eint {
        nuvoton,pins =
            <
                //0 0x0 0x5 0 /* eint0_A0 */
                //0 0xD 0x8 0 /* eint0_A13 */

                //0 0x1 0x5 0 /* eint1_A1 */
                //0 0xE 0x6 0 /* eint1_A14 */

                //3 0x0 0x4 0 /* eint2_D0 */
                //4 0xA 0x5 0 /* eint2_E10 */
                //1 0x3 0x3 0 /* eint2_B3 */
                //1 0xD 0x2 0 /* eint2_B13 */

                //3 0x1 0x4 0 /* eint3_D1 */
                //4 0xC 0x5 0 /* eint3_E12 */
                //6 0xF 0x4 0 /* eint3_G15 */

                1 0x08 0x0 0 /* PB8 */
                1 0x0D 0x0 0 /* PB13 */
                6 0x0F 0x0 0 /* PG15 */
            >;
    };
};
```

```
ccap0 {
    pinctrl_ccap0: ccap0 {
        nuvoton,pins =
```

# | Accessing GPIO from User Space

- Use GPIO from Linux shell

```
# echo 40 > /sys/class/gpio/export  
# echo out > /sys/class/gpio/gpio40/direction  
# echo 1 > /sys/class/gpio/gpio40/value  
# echo 0 > /sys/class/gpio/gpio40/value
```

# Accessing GPIO from Kernel Space

- Edit the device tree header file  
`<linux/arch/arm/boot/dts/nuc980.dtsi>`
- In the device node entry `apb:gpio:gpio@b0004000`, configure GPIO pins PB8, PB13 and PG15

```
gpios = <
    &gpio 0x28 GPIO_ACTIVE_LOW /* PB8 */
    &gpio 0x2D GPIO_ACTIVE_LOW /* PB13 */
    &gpio 0xCF GPIO_ACTIVE_LOW /* PG15 */
>;
```

```
/* 2nd: 0(PA1)/1(PA14) */
/* 3rd: 0(both)/1(rasing)/2(falling)/3(heigh level)/4(low level) */
eint1-config = <0 0 0>;

/* 1st: 0(Disable)/1(Enable) */
/* 2nd: 0(PD0)/1(PE10)/2(PB3)/3(PB13) */
/* 3rd: 0(both)/1(rasing)/2(falling)/3(heigh level)/4(low level) */
eint2-config = <0 3 0>;

/* 1st: 0(Disable)/1(Enable) */
/* 2nd: 0(PD1)/1(PE12)/2(PG15) */
/* 3rd: 0(both)/1(rasing)/2(falling)/3(heigh level)/4(low level) */
eint3-config = <0 2 0>;

gpios = <
    &gpio 0x28 GPIO_ACTIVE_LOW /* PB8 */
    &gpio 0x2D GPIO_ACTIVE_LOW /* PB13 */
    &gpio 0xCF GPIO_ACTIVE_LOW /* PG15 */
>;

status = "okay";

};

nadc: nadc@b0043000 {
    compatible = "nuvoton,nuc980-nadc";
    reg = <0xb0043000 0x100>;
    interrupts = <18 4 1>;
    map-addr = <0xf0043000>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_nadc>;
```

# | Accessing GPIO from Kernel Space

- Edit the Linux GPIO driver source file  
**<linux/drivers/gpio/gpio-nuc980.c>**
- In probe function  
**nuc980\_gpio\_probe()**,  
add the code showed on  
the right side.

## linux/drivers/gpio/gpio-nuc980.c

```
static int nuc980_gpio_probe(struct platform_device *pdev)
{
    int err, i, gpio_pin_count, gpio_pin_num;
    ...
    #ifdef CONFIG_GPIO_NUC980_EINT_WKUP
        nuc980_enable_eint(1, pdev);
    #else
        nuc980_enable_eint(0, pdev);
    #endif

    gpio_pin_count = of_gpio_count(pdev->dev.of_node);
    for (i = 0; i < gpio_pin_count; i++) {
        gpio_pin_num = of_get_gpio(pdev->dev.of_node, i);

        if (gpio_is_valid(gpio_pin_num)) {
            if (gpio_pin_num == NUC980_PG15) {
                gpio_direction_output(gpio_pin_num, 1);
            }

            if (gpio_pin_num == NUC980_PB13) {
                gpio_direction_input(gpio_pin_num);
            }
        }
    }
}
```

*Joy of innovation*  
**nuvoTon**

谢谢

謝謝

Děkuji

Bedankt

Thank you

Kiitos

Merci

Danke

Grazie

ありがとう

감사합니다

Dziękujemy

Obrigado

Спасибо

Gracias

Teşekkür ederim

Cảm ơn