GPIO PWM & Capture Linux 模組驅動

NuMicro® 64/32位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式說明如何在 Linux 核心下導入 GPIO PWM 與 GPIO Capture 驅動，實現 PWM 輸出與偵測統計腳位變化狀態。
BSP 版本	Linux-5.10.x
開發平台	NuMaker-IOT-MA35D1 V2.0

1. 概述

整體概念如圖 1-1 所示，主要有兩個部分；1. GPIO PWM：藉由 GPIO 輸出脈寬調變。2. GPIO Capture：偵測腳位輸入的狀態變化與時間等訊息。

本範例程式示範如何經由 GPIO PWM 核心模組模擬輸出 PWM 訊號，並將訊號連接至 GPIO，藉由 GPIO Capture 核心模組觸發中斷來計算中斷次數，可以透過 “Capture” 測試程式與 /sys 介面讀取中斷發生的資訊。

考量實用性，GPIO Capture 在設計上核心模組最多僅支援 32 個的腳位，但因為使用到核心裡面的高精度時間實現腳位二次偵測功能，因此；過多的腳位偵測是否影響到系統效能，這一點在使用上需要注意。

至於 GPIO PWM 訊號的數量目前並沒有限制，但因為該模塊使用高精度時間去計算佔空比，如果數量太多的話會影響到系統的效率進而影響到佔空比的精確度。

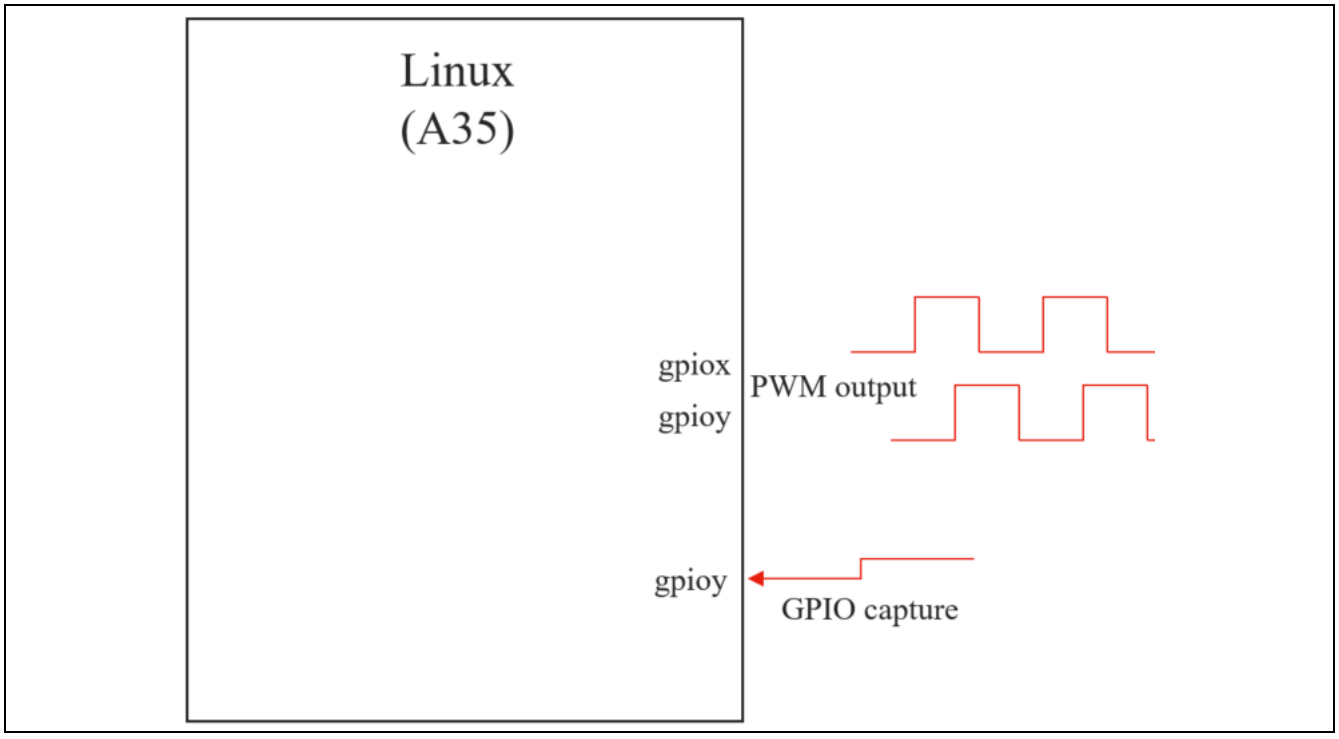

圖 1-1 功能概述

1.1 軟體架構

軟體實現架構如圖 1-2 所示，使用者可以藉由修改設備樹來設定驅動所需要的訊息，驅動程式根據使用者設定註冊 PWM 與所要偵測狀態變化的腳位，驅動程式依據使用者的設定來執行驅動。

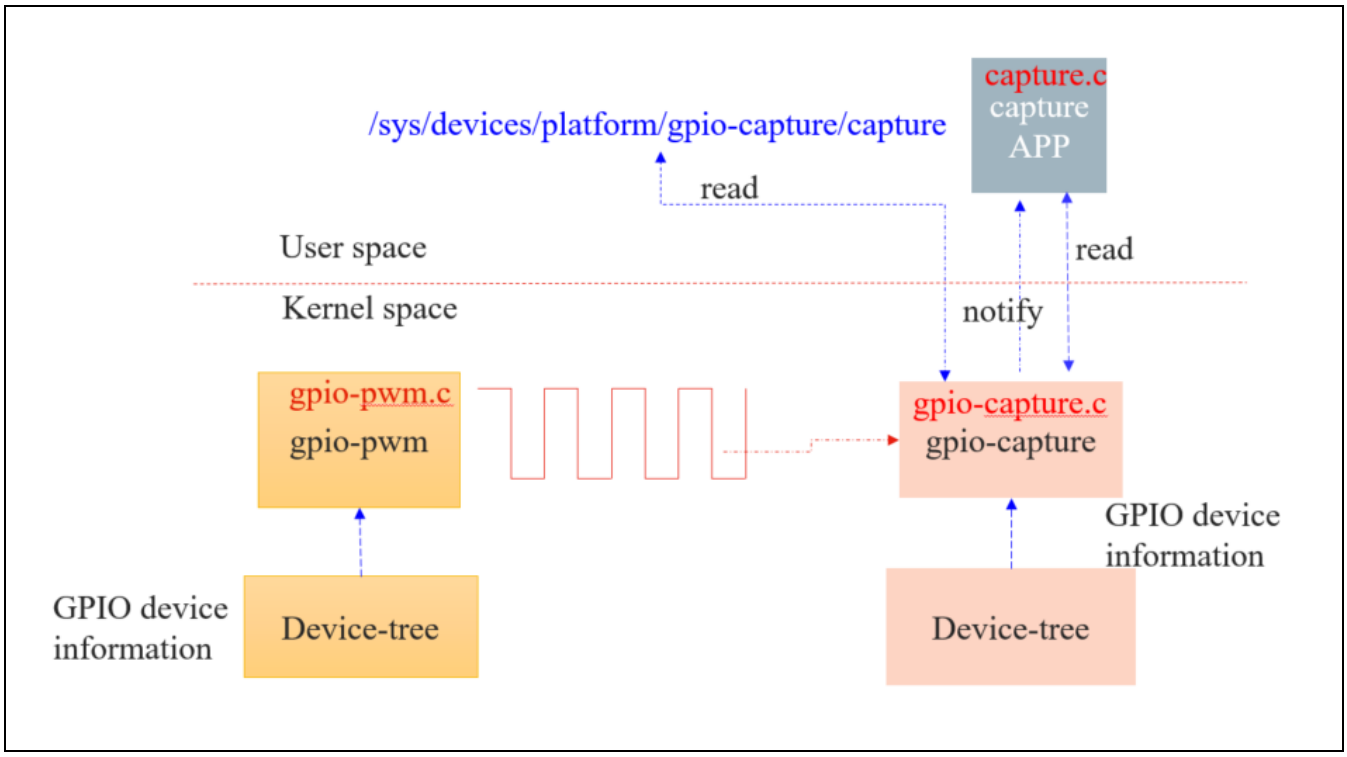

圖1-2 軟體架構圖

1.2 驅動設定

1.2.1 GPIO PWM 設定

1.2.1.1 Kernel configuration settin

GPIO PWM 必須開啟 Linux 的 PWM 編譯選項如下。

Device Drivers ---> [*] Pulse-Width Modulation (PWM) Support

1.2.1.2 設備樹設定

```
/* 1. Add node in device tree root's configuration */
/ {
    model = "Nuvoton MA35D1-IoT";
    ....
    gpio_pwm {
    compatible = "gpio-pwm";
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_gpio_pwm>;
    gpios = <&gpioi 12 GPIO_ACTIVE_LOW>,
     <&gpioi 13 GPIO_ACTIVE_LOW>;
    };
};

/* 2. Add module pin configuration to device tree's pinctl as below */
&pinctrl {
....
gpio_pwm {
        pinctrl_gpio_pwm: gpio_pwmgrp{
       nuvoton,pins =
       <SYS_GPI_MFPH_PI12MFP_GPIO &pcfg_default>,
       <SYS_GPI_MFPH_PI13MFP_GPIO &pcfg_default>;
        };
};
....
};
```

- gpio output pin：

  PWM 輸出腳位選擇，須注意 Device tree 在設定上是以 port 加上 pin，以 gpioi 12 來說明在 .c 中的設定須將 port 轉為 gpio 數，如 gpioi = 16*8 = 128, gpioi 12 即為 128+12 = 140。

- pwm chip base：

  註冊 pwm 時的 chip 索引值，目前 PWM 模擬晶片位址固定為 32，於系統註冊時會產生/sys/class/pwm/pwmchip32 的介面以供使用者操作使用。

1.2.2 GPIO Capture 設定

```
/* 1. Add node in device tree root's configuration */
/ {
    model = "Nuvoton MA35D1-IoT";
    ....
    gpio_cap {
    compatible = "gpio-capture";
    status = "okay";
    debounce-us = <20>;
    irq-type = <IRQ_TYPE_EDGE_RISING>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_gpio_cap>;
    gpios = <&gpioi 14 GPIO_ACTIVE_LOW>,
     <&gpioi 15 GPIO_ACTIVE_LOW>;
    };
};

/* 2. Add module pin configuration to device tree's pinctl as below */
&pinctrl {
....
gpio_cap {
       pinctrl_gpio_cap: gpio_capgrp{
        nuvoton,pins =
        <SYS_GPI_MFPH_PI14MFP_GPIO &pcfg_default>,
        <SYS_GPI_MFPH_PI15MFP_GPIO &pcfg_default>;
       };
};
....
};
```

- capture gpio index：capture 輸入偵測腳位選擇

  須注意 Device tree 在設定上是以 port 加上 pin，以 gpioi 14 為例在程式 .c 中的設定須將 port 轉為 gpio 數，如 gpioi = 16*8 = 128, gpioi 14 即為 128+14 = 142 。

- capture type：腳位偵測的觸發型態，目前有

  IRQ_TYPE_EDGE_RISING,

  IRQ_TYPE_EDGE_FALLING,

  IRQ_TYPE_LEVEL_HIGH,

  IRQ_TYPE_LEVEL_LOW，等幾種設定值。

- filter time

  腳位偵測確認狀態的間隔時間，以 debounce_us=20 為例，會在第一次偵測到狀態改變時，先抓一次腳位的狀態為 s1，在 20us 後會抓第二次的腳位的狀態為 s2，比較 s1 等於 s2 後才算狀態改變。

1.2.3 執行說明

1.2.3.1 GPIO PWM

GPIO PWM 模組執行後，會輸出如圖 1-3 所示的 PWM (Pulse Width Modulation 脈衝寬度調變) ，使用者可以透過 Linux 的介面調整 ”週期” 與 “佔空比” 輸出，調整方式在範例程式執行章節中介紹。

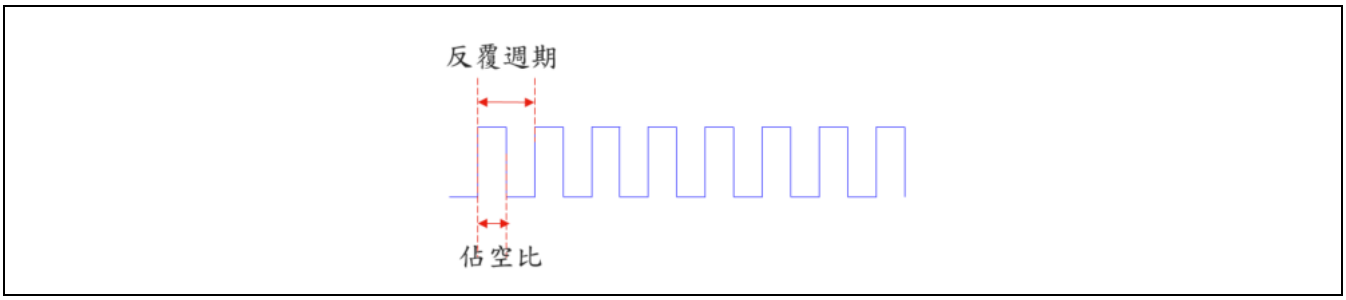

圖 1-3 PWM 輸出

1.2.3.2 GPIO Capture

GPIO Capture 模組執行後，會偵測所設定腳位的輸入變化，輸出可以經由 /sys 與使用者程式由 read 去讀取資料(圖 1-2)，輸出格式如圖 1-4 所示, 說明如下。

- state : 腳位偵測狀態。
- gpiox : 腳位編號。
- count : 偵測腳位改變次數。
- time : 腳位改變時間(系統 jiffies 低 32 位)。

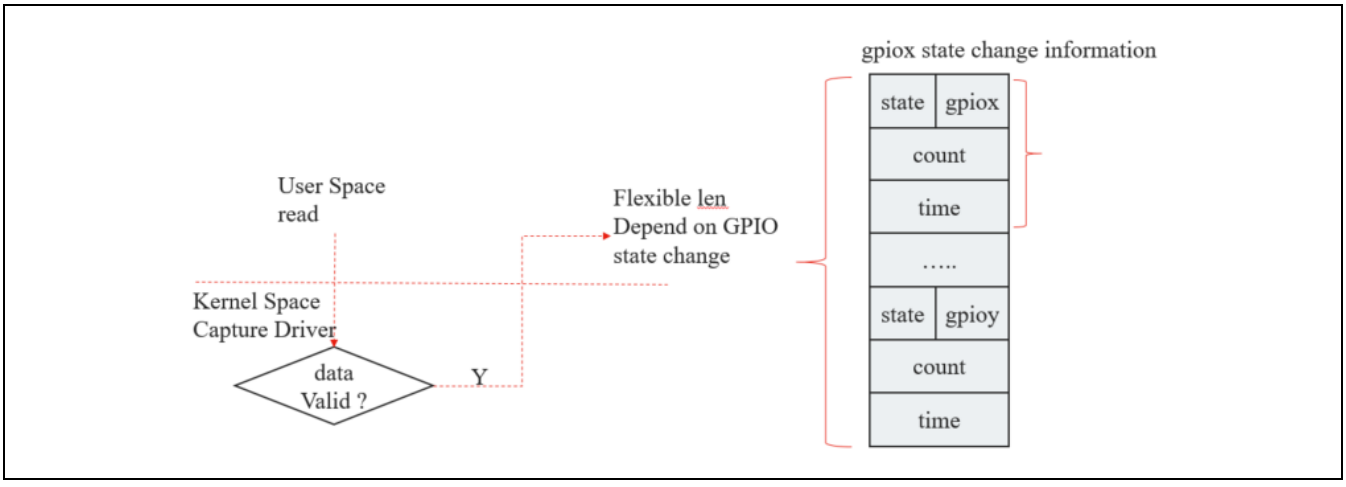

圖 1-4 capture 輸出格式

1.2.3.3 執行結果

執行下列兩個步驟後

1. 硬體接線，請參考章節 3.2 硬體需求。

2. 安裝核心模塊與執行應用端軟件，請參考章節 5 範例程式執行。

經由上述步驟後, 可以看到如下結果，輸出格式請參考章節 1.2.3.2 GPIO Capture。

```
gpio142=1  count=1228, time=10375353
gpio142=1  count=1229, time=10375363
gpio142=1  count=1230, time=10389473
gpio142=1  count=1231, time=10389491
gpio142=1  count=1232, time=10389523
```

2. 使用者程式介紹

“capture” 程式讀取 /dev/gpio_cap，輸出格式請參考章節 1.2.3.2 GPIO Capture，以下展示使用者如何透過程式方式讀取輸入腳位變化狀態資訊，說明如註解所示。

```
/* 4. If drive got gpio status change will notify user process,
this register callback function will be called */
void sig_event_handler(int sig_id, siginfo_t *sig_info, void *unused)
{
    if ( sig_id == CAP_SIG_ID ) {
        state_change = 1;
    }
}

int main()
{
    int fd;
    struct sigaction act;
    unsigned int read_buf[CAP_MAX_LEN_INT];
    int read_len;
    int data_idx;
    int line_idx;
    int i;

    /* 1. Signal callback function register */
    sigemptyset(&act.sa_mask);
    act.sa_flags = (SA_SIGINFO | SA_RESTART);
    act.sa_sigaction = sig_event_handler;
    sigaction(CAP_SIG_ID, &act, NULL);

    /* 2. Open device & in driver will catch process notify information */
    fd = open("/dev/gpio_cap", O_RDWR);
    if(fd < 0) {
        printf("Open device fail\n");
        return 0;
    }

    int gpio;
    int state;
    unsigned int count;
    unsigned int time;
    int gpio_info_nums;
    unsigned int pre_count;

    while(1) {
        fflush(0);

    /* 3. check if drive got new gpio status */
        if (state_change == 1) {
    /* 5. Got gpio status change */
            read_len = read(fd, read_buf, CAP_MAX_LEN);
            read_len = read_len / 4;
            gpio_info_nums = read_len / GPIO_INFO_SIZE_INT;   /* 3 int gpio info */
```

```
        /* 6. Read all change gpio status information */
             for (i=0; i<gpio_info_nums; i++) {
                 pre_count = count;
                 gpio = read_buf[i*GPIO_INFO_SIZE_INT] & 0x000000ff;
                 state = read_buf[i*GPIO_INFO_SIZE_INT] >> 16;
                 count = read_buf[i*GPIO_INFO_SIZE_INT + 1];
                 time = read_buf[i*GPIO_INFO_SIZE_INT + 2];

                 if ( count - pre_count != 1) {
                     printf("D");
                 }
                printf("gpio%d=%d  count=%d, time=%u\n", gpio, state, count, time);
                 }
                 state_change = 0;
          }
     }

     close(fd);
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - Linux-5.10.x
- 模組名稱如下
  - gpio-pwm.ko , gpio-capture.ko
- 應用程式 : capture

3.2 硬體需求

- 電路元件
  - NuMaker-IOT-MA35D1 V2.0
- 測試示意圖 3-1

In NuMaker-IOT-MA35D1 V2.0 Board , CON4

Select gpioi12 (gpio140) as PWM output pin

Select gpioi14 (gpio142) as Input capture pin

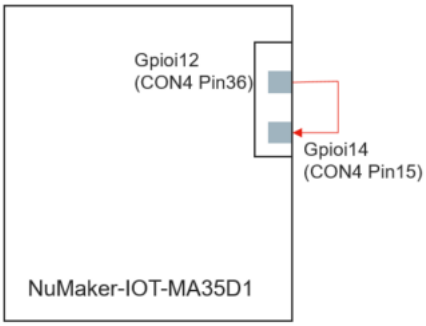

圖 3-1 測試接線圖

4. 目錄資訊

EC_MA35D1_ GPIO_Simulate_PWM_Capture_V1.00	
Src	Source files of device driver
ko	Kernel object file of device driver
capture_test	The test program of gpio status change
DT_Binding	Device tree requirements on the contents of driver

圖 4-1 目錄資訊

5. 範例程式執行

1. 編修所需的腳位(請參考章節 1.2 驅動設定)
2. 在編譯環境中重新核心與編譯核心模塊並燒錄至 MA35D1 中
3. 將 MA35D1 開機
4. 複製核心模塊與測試程式 ”capture” 到板子端
5. 導入核心模組

```
# insmod gpio-pwm.ko
# insmod gpio-capture.ko
```

6. GPIO Capture 執行腳位狀態偵測, 請參考章節 2 使用者程式介紹

```
# ./capture &
```

7. GPIO PWM 執行輸出與佔空比設定

- Export PWM

```
# echo 0 > /sys/class/pwm/pwmchip32/export
```

- PWM 週期設定,單位為 ns, gpioi12 輸出頻率為 100 HZ

```
# echo 10000000 > /sys/class/pwm/pwmchip32/pwm0/period
```

- PWM 佔空比設定, 以下設定佔空比為 0.5

```
# echo 5000000 > /sys/class/pwm/pwmchip32/pwm0/duty_cycle
```

- PWM 使能輸出

```
# echo 1 >/sys/class/pwm/pwmchip32/pwm0/enable
```

6. 修訂紀錄

Date	Revision	Description
2022.09.27	1.00	初始發布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.