GPIO PWM & Capture Linux 模块驱动

NuMicro® 64/32位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码说明如何在 Linux 核心下导入 GPIO PWM 与 GPIO Capture 驱动，实现 PWM 输出与侦测统计引脚变化状态。
BSP 版本	Linux-5.10.x
开发平台	NuMaker-IOT-MA35D1 V2.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

整体概念如图 1-1 所示，主要有两个部分；1. GPIO PWM：藉由 GPIO 输出脉宽调变。2. GPIO Capture：侦测引脚输入的状态变化与时间等信息。

本范例代码示范如何经由 GPIO PWM 核心模块仿真输出 PWM 信号，并将信号连接至 GPIO，藉由 GPIO Capture 核心模块触发中断来计算中断次数，可以透过 “Capture” 测试程序与 /sys 接口读取中断发生的信息。

考虑实用性，GPIO Capture 在设计上核心模块最多仅支持 32 个的引脚，但因为使用到核心里面的高精度时间实现引脚二次侦测功能，因此

, 过多的引脚侦测是否影响到系统效能，这一点在使用上需要注意。

至于 GPIO PWM 信号的数量目前并没有限制，但因为该模块使用高精度时间去计算占空比，如果数量太多的话会影响到系统的效率进而影响到占空比的精确度。

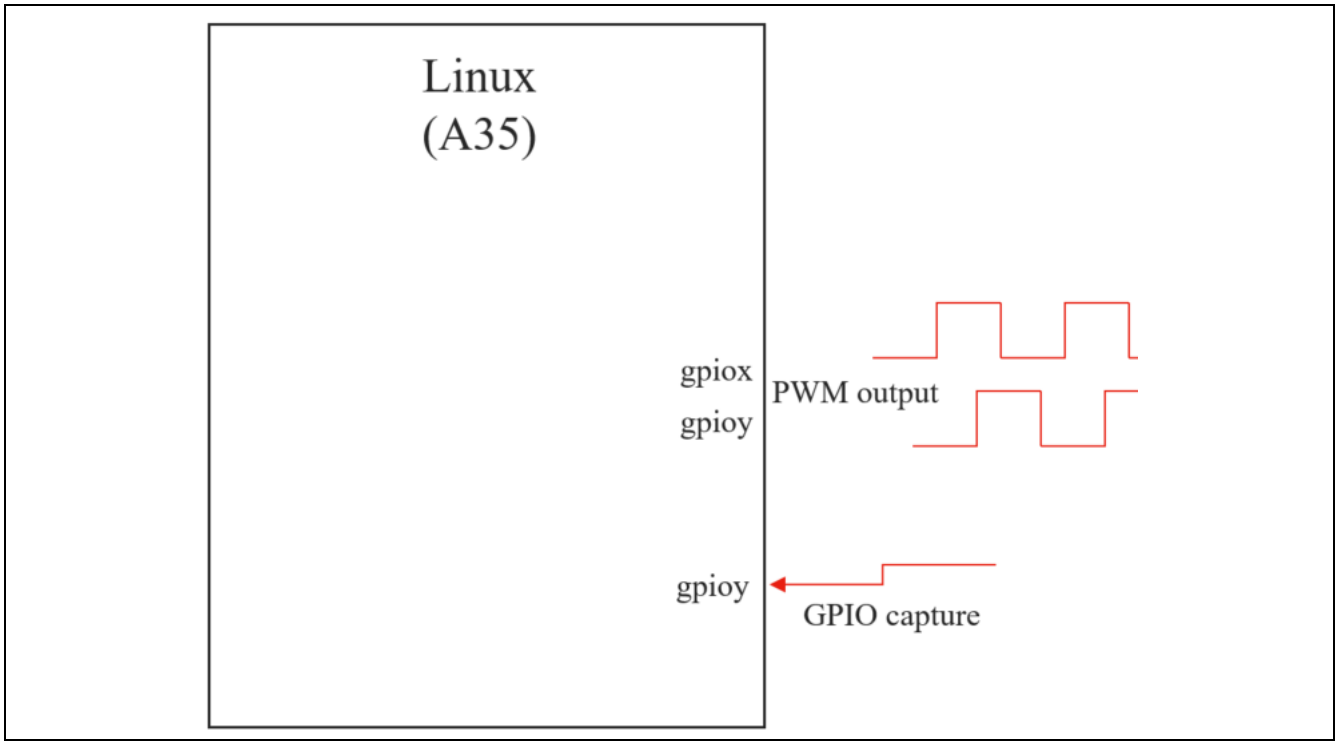

图 1-1 功能概述

1.1 软件架构

软件实现架构如图 1-2 所示，用户可以藉由修改设备树配置來设定驱动所需要的信息，驱动程序根据用户设定注册 PWM 与所要侦测状态变化的引脚，驱动程序依据用户的设定来执行驱动。

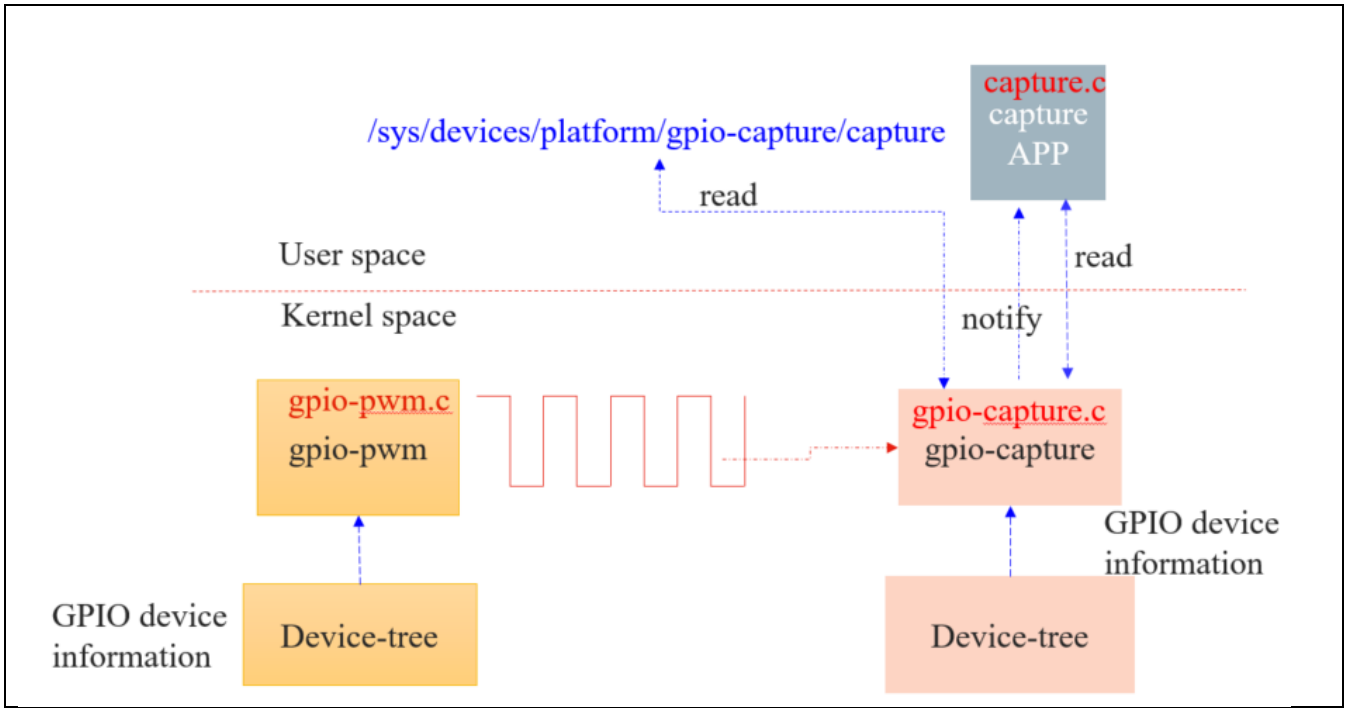

图 1-2 软件架构图

1.2 驱动设定

1.2.1 GPIO PWM 设定

1.2.1.1 内核编译设定

GPIO PWM 必须开启 Linux 的 PWM 编译选项如下。

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

1.2.1.2 设备树设定

```
/* 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 输出引脚选择，须注意 Device tree 在设定上是以 port 加上 pin，以 gpioi 12 来说明在 .c 中的设定须将 port 转为 gpio 数，如 gpioi = 16*8 = 128, gpioi 12 即为 128+12 = 140 。

- PWM chip base：

  注册 pwm 时的 chip 索引值，目前 PWM 仿真芯片地址固定为 32，于系统注册时会产生 /sys/class/pwm/pwmchip32 的界面以供用户操作使用。

1.2.2 GPIO Capture 设定

```
/* 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 输入侦测引脚选择

  须注意 Device tree 在设定上是以 port 加上 pin，以 gpioi 14 为例在程序 .c 中的设定须将 port 转为 gpio 数，如 gpioi = 16*8 = 128, gpioi 14 即为 128+14 = 142 。

- capture type：引脚侦测的触发型态，目前有

  IRQ_TYPE_EDGE_RISING,

  IRQ_TYPE_EDGE_FALLING,

  IRQ_TYPE_LEVEL_HIGH,

  IRQ_TYPE_LEVEL_LOW，等几种设定值。

- filter time

  脚位侦测确认状态的间隔时间，以 debounce_us=20 为例，会在第一次侦测到状态改变时，先抓一次引脚的状态为 s1，在 20us 后会抓第二次的引脚的状态为 s2，比较 s1 等于 s2 后才算状态改变。

1.2.3 执行说明

1.2.3.1 GPIO PWM

GPIO PWM 模块执行后，会输出如图 1-3 所示的 PWM（Pulse Width Modulation 脉冲宽度调变），用户可以透过 Linux 的接口调整 ”周期” 与 “占空比” 输出，调整方式在范例程序执行章节中介绍。

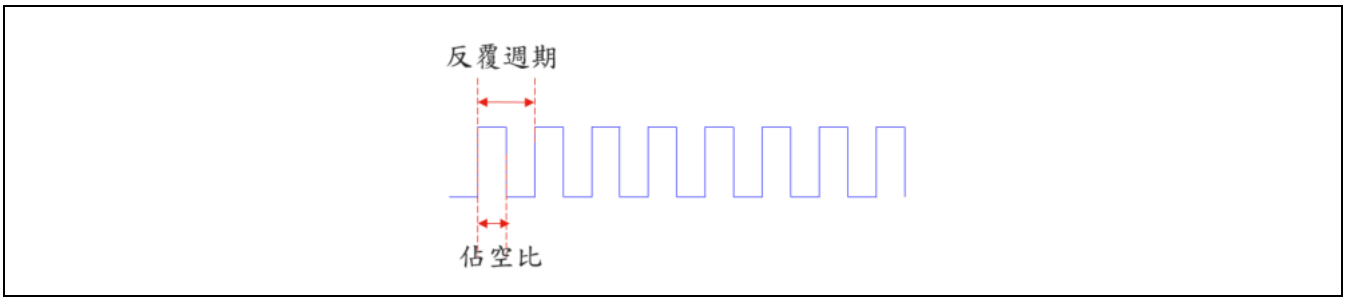

图 1-3 PWM 输出

1.2.3.2 GPIO Capture

GPIO Capture 模块执行后，会侦测所设定引脚的输入变化，输出可以经由 /sys 与用户程序由 read 去读取数据(图 1-2)，输出格式如图 1-4 所示, 说明如下。

- state : 引脚侦测状态。
- gpiox : 引脚编号。
- count : 侦测引脚改变次数。
- time : 引脚改变时间(系统 jiffies 低 32 位)。

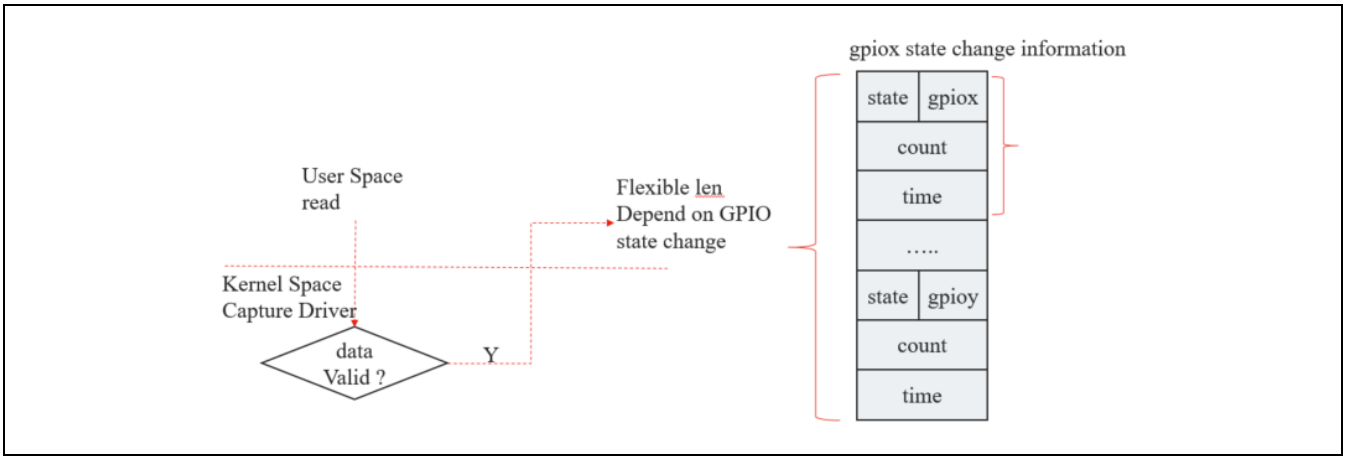

图 1-4 capture 输出格式

1.2.3.3 执行结果

执行下列两个步骤后

1. 硬件接线，请参考章节 3.2 硬件需求。
2. 安装核心模块与执行应用端软件，请参考章节 5 范例程序执行。

经由上述步骤后, 可以看到如下结果，输出格式请参考章节 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. 使用者代码介绍

“capture” 程序读取 /dev/gpio_cap，输出格式请参考章节 1.2.3.2 GPIO Capture，以下展示使用者如何透过代码方式读取输入引脚变化状态信息，说明如批注所示。

```
/* 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. 软件与硬件需求

3.1 软件需求

- BSP 版本
- Linux-5.10.x
- 模块名称如下
- gpio-pwm.ko , gpio-capture.ko
- 应用程序 : capture

3.2 硬件需求

- 电路组件
- NuMaker-IOT-MA35D1 V2.0
- 测试示意圖

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

Select gpioi12 (gpio140) as PWM output pin

Select gpioi14 (gpio142) as Input capture pin

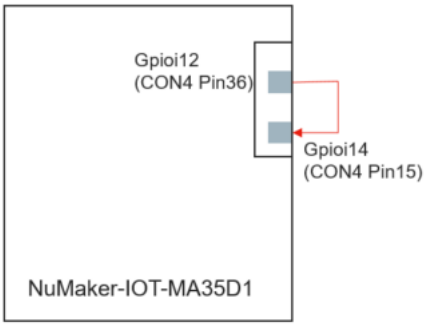

图 3-1 测试接线图

4. 目录信息

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

图 4-1 目录信息

5. 范例程序执行

1. 编修所需的引脚(请参考章节 1.2 驱动设定)
2. 在编译环境中重新编译核心模块并烧录至 MA35D1 中
3. 将 MA35D1 开机
4. 复制核心模块与测试程序 ”capture” 到板子端
5. 导入核心模块

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

6. GPIO Capture 执行引脚状态侦测, 请参考章节 2 使用者代码介绍

```
# ./capture &
```

7. GPIO PWM 执行输出与占空比设定
   - Export pwm

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

   - PWM 周期设定,单位为 ns, gpioi12 输出频率为 100 HZ

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

   - PWM 占空比设定, 以下设定占空比为 0.5

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

   - PWM 使能输出

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

6. 修订纪录

Date	Revision	Description
2022.09.27	1.00	初始发布。

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.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.