SPI转CAN第一次启动报错问题

前言

- 本项目中RK3399PRO由于没有原生can控制器，因此需要用到转接芯片；本次使用的mcp251x芯片，SPI转CAN；

报错信息描述

- 使用的can启动设置命令如下

```
ip link set can0 down
ip link set can0 type can bitrate 500000
ip link set can0 up
```

- 执行到第三条也就是can0 up的时候会报错

```
root@bionic:~# ip link set can0 up
RTNETLINK answers: Invalid argument
```

解决思路

- 应用层使用ip命令最终也是会调用到驱动中的相关函数，既然如此，那就去查一下驱动吧，可以看见注册了一个net_device_ops操作方法结构体，这个结构体中函数指针指向的函数就是上层操作的时候调用的函数，当执行**ip link set can0 up**的时候就会去调用ops结构体中的open函数，所以猜测问题肯定出在**mcp251x_open**这个函数中；

```
static const struct net_device_ops mcp251x_netdev_ops = {
    .ndo_open = mcp251x_open,
    .ndo_stop = mcp251x_stop,
    .ndo_start_xmit = mcp251x_hard_start_xmit,
    .ndo_change_mtu = can_change_mtu,
};
```

- 如果直接波特率都不设置呢，执行会有什么结果呢？果然，会报刚才的报错，查看一下打印log，提示是波特率没有设置，所以这里可以得出初步结论**net_device_ops**中的open指针指向的函数如果返回的不是0，那么就会报这个错误；

```
[  109.168613] === func: mcp251x_open net ops mcp251x open! ===
[  109.169138] mcp251x spi32766.0 can0: bit-timing not yet defined
[  109.169660] mcp251x spi32766.0: unable to set initial baudrate!
[  111.024283] RTW: _lps_chk_by_tp(wlan0) tx_tp:0 [12], rx_tp:0 [12], bi_tp:0 [12], en
```

```c
static int mcp251x_open(struct net_device *net)
{
    struct mcp251x_priv *priv = netdev_priv(net);
    struct spi_device *spi = priv->spi;
    unsigned long flags = IRQF_ONESHOT | IRQF_TRIGGER_FALLING;
    int ret;

    ret = open_candev(net);
    if (ret)
    {
        dev_err(&spi->dev, "unable to set initial baudrate!\n");
        return ret;
    }
```

- 为了验证这个想法，可以执行在open多加几条打印函数，看看会卡在哪里，按照步骤复现之后果然第一次执行并没有完整的走完整个open函数，所以问题已经很明确了，定位到执行出错的函数查看是什么原因导致的错误。

```
root@bionic:~# ip link set can0 type can bitrate 500000
root@bionic:~# ip link set can0 up
[   84.983651] === func: mcp251x_open, line:940 net ops mcp251x open! ===
[   84.984675] === func: mcp251x_open, line:963 net ops mcp251x open! ===
RTNETLINK answers: No such device
root@bionic:~# ip link set can0 up
[   87.201338] === func: mcp251x_open, line:940 net ops mcp251x open! ===
[   87.202534] === func: mcp251x_open, line:963 net ops mcp251x open! ===
[   87.213868] === func: mcp251x_open, line:984 net ops mcp251x open! ===
[   87.214461] === func: mcp251x_open, line:988 net ops mcp251x open! ===
```

- 通过继续加打印，定位到问题出在下图所示的函数中，在这个函数中，读取can的状态出了问题，那为什么第二次读取的时候就能正常读取呢？

```c
static int mcp251x_hw_reset(struct spi_device *spi)
{
    struct mcp251x_priv *priv = spi_get_drvdata(spi);
    u8 reg;
    int ret;

    /* Wait for oscillator startup timer after power up */
    mdelay(MCP251X_OST_DELAY_MS);

    priv->spi_tx_buf[0] = INSTRUCTION_RESET;
    ret = mcp251x_spi_trans(spi, 1);
    if (ret)
        return ret;

    /* Wait for oscillator startup timer after reset */
    mdelay(MCP251X_OST_DELAY_MS);

    reg = mcp251x_read_reg(spi, CANSTAT);
    if ((reg & CANCTRL_REQOP_MASK) != CANCTRL_REQOP_CONF)
        return -ENODEV;

    return 0;
}
```

1 继续往下追是这里出了问题

- 通过打印发现读取的值第一次是0xff显然不正确，再次执行指令读取到的值是0x80为什么会出现这种情况呢？是spi出了问题还是这个芯片第一次读取数据有问题呢？

```
root@bionic:~# ip link set can0 up
[   58.275345] === func: mcp251x_hw_reset line: 648 reg: 0xff ===
RTNETLINK answers: No such device
root@bionic:~# ip link set can0 up
[   79.711635] === func: mcp251x_hw_reset line: 648 reg: 0x80 ===
```

- 由于spi是soc原生控制器直出的问题的概率很低，所以排查方向优先放在mcp251x芯片上；在出问题的这个函数中，是对芯片做了一个复位操作，并且等待5ms之后去读复位状态，那么是不是问题就出在这个等待时间上，查看datasheet中并没有相关的定义，尝试加长延时时间到15ms；

```
    /* Wait for oscillator startup timer after reset */
    mdelay(MCP251X_OST_DELAY_MS * 3);
    reg = mcp251x_read_reg(spi, CANSTAT);
    if ((reg & CANCTRL_REQOP_MASK) != CANCTRL_REQOP_CONF)
        return -ENODEV;

    return 0;
```

- 烧录之后再次测试问题解决！！！