

## 驱动介绍文档

### 一、开发环境介绍

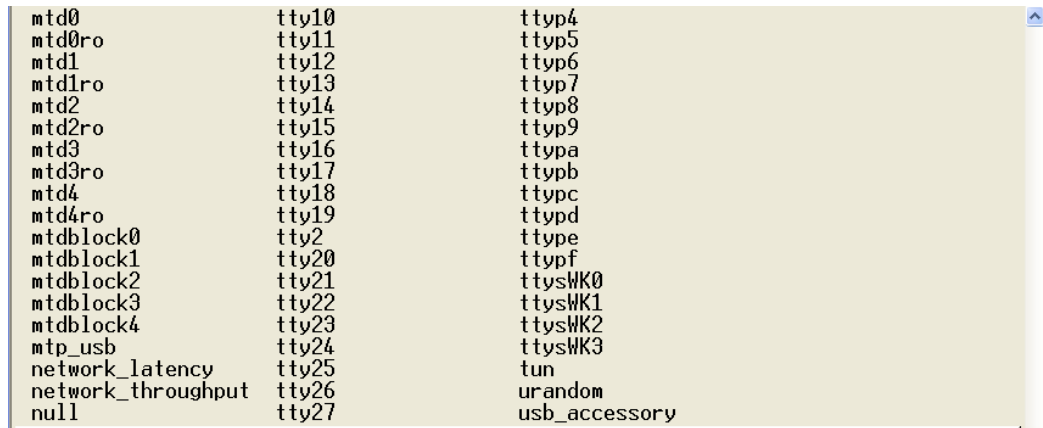
#### 开发平台说明

本驱动是在友善之臂的 tiny210 的平台上开发和调试的。Tiny210 使用了三星的 S5PV210 作为主控制器，我们使用的内核的版本是 Liunx-3.0.8。

### 二、驱动移植注意事项

- 1、由于我们的驱动的主 UART 是直接操作 cpu 的相关寄存器。
- 2、函数 void wk\_s5pv210uart\_InitIO(void) 主要实现 cpu 的 io 初始化和 IO 内存映射。这个函数根据不同硬件实现方式不一样。你也可以调用你已有的 UART 驱动来实现。
- 3、由于 UART 的波特率是自适应的，也就是不需要用去配置我们芯片的主接口 uart 的波特率，用户只需要在芯片复位后，向我们芯片发送一个 0x55,wk2xxx 就会自动测得此时 cpu 的波特率，并自动把自身的波特率锁定为此波特率，以后 cpu 和 wk2xxx 只能按照锁定的波特率进行通信。如果要改变 cpu 和 wk2xxx 通信的波特率，只有对芯片进行硬件复位，所以如果要使用 wk2xxx 的 uart 扩展 uart 功能，建议在硬件设计的时候考虑用 IO 控制一下 wk2xxx 的复位引脚。

4、驱动注册成功的设备名，如下图：ttysWK0, ttysWK1, ttysWK2, ttysWK3,



mtd0	ttty10	tttyp4
mtd0ro	ttty11	tttyp5
mtd1	ttty12	tttyp6
mtd1ro	ttty13	tttyp7
mtd2	ttty14	tttyp8
mtd2ro	ttty15	tttyp9
mtd3	ttty16	tttypa
mtd3ro	ttty17	tttypb
mtd4	ttty18	tttypc
mtd4ro	ttty19	tttypd
mtdblock0	ttty2	tttype
mtdblock1	ttty20	tttypf
mtdblock2	ttty21	tttysWK0
mtdblock3	ttty22	tttysWK1
mtdblock4	ttty23	tttysWK2
mtp_usb	ttty24	tttysWK3
network_latency	ttty25	tun
network_throughput	ttty26	urandom
null	ttty27	usb_accessory

- 5、波特率设置：由于可能使用的晶振不一样，所以我们可能要修改下面的函数，以便实现晶振和波特率实现匹配也就是我们能正确的配置 baud1,baud0,pres 这 3 个寄存器，我们调试时使用的是 11.0592 的波特率。

```
static void wk2xxx_termios( struct uart_port *port, struct ktermios *termios,
                           struct ktermios *old)
{
#ifdef _DEBUG_WK2XXX
    printk(KERN_ALERT "-vk32xx_termios-----in---\n");
#endif

    struct wk2xxx_port *s = container_of(port,struct wk2xxx_port,port);
    int baud = 0;
    uint8_t lcr,scr,baud1,baud0,pres;
    unsigned short cflag;
    unsigned short lflag;
```

```

//u32 param_new, param_mask;
cflag = termios->c_cflag;
lflag = termios->c_lflag;
#ifdef _DEBUG_WK2XXX
    printk(KERN_ALERT "cflag := 0x%X   lflag : = 0x%X\n",cflag,lflag);
#endif
    baud1=0;
    baud0=0;
    pres=0;
    baud = tty_termios_baud_rate(termios);
    //sysclk=11.0592MHZ
    switch (baud) {
    case 600:
        baud1=0x40;
        baud0=0x7f;
        pres=0;
        break;
    case 1200:
        baud1=0x20;
        baud0=0x3f;
        pres=0;
        break;
    case 2400:
        baud1=0x10;
        baud0=0x1f;
        pres=0;
        break;
    case 4800:
        baud1=0x00;
        baud0=0x8f;
        pres=0;
        break;
    case 9600:
        baud1=0x00;
        baud0=0x47;
        pres=0;
        break;
    case 19200:
        baud1=0x00;
        baud0=0x23;
        pres=0;
        break;
    case 38400:
        baud1=0x00;

```

```
        baud0=0x11;
        pres=0;
    break;
case 76800:
    baud1=0x00;
    baud0=0x05;
    pres=0;
    break;

case 1800:
    baud1=0x01;
    baud0=0x7f;
    pres=0;
    break;
case 3600:
    baud1=0x00;
    baud0=0xbf;
    pres=0;
    break;
case 7200:
    baud1=0x00;
    baud0=0x5f;
    pres=0;
    break;
case 14400:
    baud1=0x00;
    baud0=0x2f;
    pres=0;
    break;
case 28800:
    baud1=0x00;
    baud0=0x17;
    pres=0;
    break;
case 57600:
    baud1=0x00;
    baud0=0x0b;
    pres=0;
    break;
case 115200:
    baud1=0x00;
    baud0=0x05;
    pres=0;
    break;
```

```

case 230400:
    baud1=0x00;
    baud0=0x02;
    pres=0;
    break;
default:
    baud1=0x00;
    baud0=0x00;
    pres=0;
}
tty_termios_encode_baud_rate(termios, baud, baud);

/* we are sending char from a workqueue so enable */

//spin_lock(&s->conf_lock);

//s->port.state->port.tty->low_latency = 1;
//termios->c_lflag &= ~ECHO;

lcr =0;
if (cflag & CSTOPB)
    lcr|=WK2XXX_STPL;//two  stop_bits
else
    lcr&=~WK2XXX_STPL;//one  stop_bits

if (cflag & PARENB) {
    lcr|=WK2XXX_PAEN;//enable spa
    if (!(cflag & PARODD)){
        lcr |= WK2XXX_PAM0;//PAM0=1
        lcr &= ~WK2XXX_PAM0;//PAM1=0
    }
    else{
        lcr |= WK2XXX_PAM1;
        lcr &= ~WK2XXX_PAM0;
    }
}
else{
    lcr&=~WK2XXX_PAEN;
}

//scr = 0;
//scr &= 0x0f;
//scr |= param_new<<4;
s->new_baud1=baud1;

```

```

s->new_baud0=baud0;
s->new_pres=pres;

//spin_lock(&s->conf_lock);
//s->conf_flag =1;
s->new_lcr = lcr;
s->new_scr = scr;
//spin_unlock(&s->conf_lock);
//vk32xx_dowork(s);

if(!(s->conf_flag|| s->conf_fail))
{
if(wk2xxx_dowork(s))
{
s->conf_flag =1;
}
else
{
s->conf_fail =1;
}

}

#ifdef _DEBUG_WK2XXX
printk(KERN_ALERT "-vk32xx_termios-----exit---\n");
#endif

return ;

}

```

## 6、修改驱动注意事项

由于我们的芯片的寄存器分布为 PAGE0 和 PAGE1 上。我们在驱动当中把 PAGE0 作为默认的页，如果要写 PAGE1 上的寄存器，我们先转换到 PAGE1 上，然后操作相应的寄存器，操作完以后再回到默认页 PAGE0 上，这样可以避免页混乱。