iTOP-4412-驱动-usb 文档 07-鼠标驱动完结

本文档中，在驱动中添加输入子系统相关的部分代码，在鼠标左键，右键等操作之后，能够打印对应的信息。

1 输入子系统初始化

在前面，已经学习过输入子系统，这里就不再重复，代码中有详细的注释，如下所示。

```
static int usb_mouse_probe(struct usb_interface *intf, const struct usb_device_id *id)
{
	struct usb_device *dev = interface_to_usbdev(intf);
	struct usb_endpoint_descriptor *endpoint;
	int ret;
	//鉴于我们在中断处理函数中，需要监听是否有数据传输，这几个变量设置为 static 变量
#if 0
	int pipe, maxp;
	signed char *data;
	dma_addr_t data_dma;
	struct urb *mouse_urb;*/
#endif

	printk("usb mouse probe!\n");
	//check_usb_device_descriptor(dev);

	//申请 input_dev
	input_dev = input_allocate_device();

	//设置  能产生哪类事件和具体的哪些时间
	set_bit(EV_KEY, input_dev->evbit);
	set_bit(EV_REP, input_dev->evbit);

	set_bit(KEY_A, input_dev->keybit);
	set_bit(KEY_B, input_dev->keybit);
	//注册输入子系统
	ret = input_register_device(input_dev);
```

```
    //在 usb 的数据传输中，"数据源"是端点， “传输通道"是管道，"数据终端"是内核 “USB 驱动” 中的
buffer
    //数据源，数据通道，数据终端全部是在 urb 中设置
    endpoint = &intf->cur_altsetting->endpoint[0].desc;      //数据源

    pipe = usb_rcvintpipe(dev, endpoint->bEndpointAddress);    //传输通道
    maxp = usb_maxpacket(dev, pipe, usb_pipeout(pipe));

    data = usb_alloc_coherent(dev, 8, GFP_ATOMIC, &data_dma);//数据终端

    //urb 必须使用内核函数申请，以便内核同一管理
    mouse_urb = usb_alloc_urb(0, GFP_KERNEL);
    //通过 urb 设置数据源，传输通道，数据终端
    usb_fill_int_urb(mouse_urb, dev, pipe, data,(maxp > 8 ? 8 : maxp),check_usb_data, NULL, endpoint-
>bInterval);
    mouse_urb->transfer_dma = data_dma;
    mouse_urb->transfer_flags |= URB_NO_TRANSFER_DMA_MAP;

    //提交 urb
    usb_submit_urb(mouse_urb, GFP_KERNEL);

    return 0;
}
```

在 probe 函数中，申请输入子系统，设置能产生哪类事件和具体的哪些事件，注册输入子系统。我们设置了键盘 key 类事件，能够产生 KEY_A 和 KEY_B 事件。

2 中断处理函数

在前一节式样中，我们已经成功实现了，鼠标操作进入中断处理函数。这里我们将采集数据和输入子系统对应，具体代码如下。

```
static void check_usb_data(struct urb *urb)
{
    unsigned char val = 0;

    //printk("count is %d time!\n",++count);

    printk("check_usb_data!\n");
    //USB 鼠标数据含义 data[0]: bit0  左键, 1 按下, 0 松开;bit1 右键,1 按下,0 松开
```

```
    //按下之后，val 获取按键值
    val = data[0];
    if(val == 0x01){
        //左键
        input_report_key(input_dev, KEY_A, 1);
        //通知上层，本次事件结束; KEY_A=30
        input_sync(input_dev);

        input_report_key(input_dev, KEY_A, 0);
        //通知上层，本次事件结束
        input_sync(input_dev);
    }
    if(val == 0x02){
        //右键
        input_report_key(input_dev, KEY_B, 1);
        //通知上层，本次事件结束;KEY_B=48
        input_sync(input_dev);

        input_report_key(input_dev, KEY_B, 0);
        //通知上层，本次事件结束
        input_sync(input_dev);
    }
    val = 0;

    //重新提交 urb
    usb_submit_urb(mouse_urb, GFP_KERNEL);
}
```

鼠标左键会打印 A，鼠标右键打印 B，然后其它鼠标操作会打印“check_usb_data!”。这部分要和后面的实验结果对照。

3 usb_mouse_disconnect

断开 USB 鼠标之后，需要添加释放输入子系统的代码，如下所示。

```
static void usb_mouse_disconnect(struct usb_interface *intf)
{
    printk("usb mouse disconnect!\n");
```

```
    usb_kill_urb(mouse_urb);
    usb_free_urb(mouse_urb);
    input_unregister_device(input_dev);
    input_free_device(input_dev);
    usb_free_coherent(interface_to_usbdev(intf), 8, data,data_dma);
}
```

4 测试

驱动编译之后，加载驱动，接上鼠标，如下图所示。

```
[root@iTOP-4412]# cd /mnt/nfs/
[root@iTOP-4412]# insmod my_usb_mouse.ko
[ 2359.704600] usbcore: registered new interface driver usbmouse
[root@iTOP-4412]# [ 2362.835255] usb 1-3.1: new low speed USB device number 4 using s5p-ehci
[ 2362.950145] usb 1-3.1: New USB device found, idVendor=046d, idProduct=c077, bcdDevice=7200
[ 2362.957141] usb 1-3.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 2362.964332] usb 1-3.1: New USB device Class: Class=0, SubClass=0, Protocol=0
[ 2362.971353] usb 1-3.1: Product: USB Optical Mouse
[ 2362.976033] usb 1-3.1: Manufacturer: Logitech
[ 2362.991182] usb mouse probe!
[ 2363.000758] input: Unspecified device as /devices/virtual/input/input3
```

操作鼠标，会打印如下图所示的信息。

```
[root@iTOP-4412]# [ 2381.443249] check_usb_data!
[ 2381.459242] check_usb_data!
[ 2381.467253] check_usb_data!
[ 2381.475250] check_usb_data!
[ 2381.483236] check_usb_data!
[ 2381.491255] check_usb_data!
[ 2381.499247] check_usb_data!
[ 2381.507251] check_usb_data!
[ 2381.515238] check_usb_data!
[ 2381.523250] check_usb_data!
[ 2381.531250] check_usb_data!
[ 2381.539250] check_usb_data!
[ 2381.547232] check_usb_data!
[ 2381.555251] check_usb_data!
[ 2381.563248] check_usb_data!
[ 2381.571249] check_usb_data!
[ 2381.579233] check_usb_data!
[ 2381.587256] check_usb_data!
[ 2382.907250] check_usb_data!
[ 2383.043244] check_usb_data!
```

接着在控制台使用命令“hexdump /dev/input/event”，鼠标左键和右键打印如下图所示信息。

```
[root@iTOP-4412]# hexdump /dev/input/event2
[ 2453.899239] check_usb_data!
0000000 0995 0000 be04 000d 0001 001e 0001 0000
0000010 0995 0000 be18 000d 0000 0000 0000 0000
0000020 0995 0000 be32 000d 0001 001e 0000 0000
0000030 0995 0000 be39 000d 0000 0000 0000 0000
[ 2454.011248] check_usb_data!
[ 2454.843247] check_usb_data!
0000040 0996 0000 e348 000c 0001 0030 0001 0000
0000050 0996 0000 e35c 000c 0000 0000 0000 0000
0000060 0996 0000 e377 000c 0001 0030 0000 0000
0000070 0996 0000 e37d 000c 0000 0000 0000 0000
[ 2454.971248] check_usb_data!
[ 2455.579250] check_usb_data!
```

如上图所示，打印了 0x1e 和 0x30，转化为十进制为 30 和 48，如下图所示，KEY_A 和 KEY_B 的数值为 30 和 48，和驱动代码中的设置一一对应。

```
#define KEY_LEFTCTRL        29
#define KEY_A               30
#define KEY_S               31
#define KEY_D               32
#define KEY_X               45
#define KEY_C               46
#define KEY_V               47
#define KEY_B               48
#define KEY_N               49
#define KEY_M               50
#define KEY_COMMA           51
#define KEY_DOT             52
```

联系方式

北京迅为电子有限公司致力于嵌入式软硬件设计，是高端开发平台以及移动设备方案提供商；基于多年的技术积累，在工控、仪表、教育、医疗、车载等领域通过 OEM/ODM 方式为客户创造价值。

iTOP-4412 开发板是迅为电子基于三星最新四核处理器 Exynos4412 研制的一款实验开发平台，可以通过该产品评估 Exynos 4412 处理器相关性能，并以此为基础开发出用户需要的特定产品。

本手册主要介绍 iTOP-4412 开发板的使用方法，旨在帮助用户快速掌握该产品的应用特点，通过对开发板进行后续软硬件开发，衍生出符合特定需求的应用系统。

如需平板电脑案支持，请访问迅为平板方案网“http://www.topeet.com”，我司将有能力为您提供全方位的技术服务，保证您产品设计无忧！

本手册将持续更新，并通过多种方式发布给新老用户，希望迅为电子的努力能给您的学习和开发带来帮助。

迅为电子

2018 年 2 月