



华为数据卡嵌入式Linux及Android内核驱动 集成指导文档

文档版本 07

发布日期 2012-12-10

华为技术有限公司为客户提供全方位的技术支持，用户可与就近的华为办事处联系，也可直接与公司总部联系。

华为技术有限公司

深圳市龙岗区坂田华为总部办公楼 公司总机：0755-28780808 网址：www.huawei.com

消费者服务热线：8008308300 4008308300 4006902116 服务邮箱：mobile@huawei.com

更多信息请访问：<http://www.huaweidevice.com>

图片仅供参考，请以实物为准。华为公司保留对产品外观及设计改进和改变的权利，恕不另行通知。


版权所有 © 华为技术有限公司 2012。保留一切权利。

非经华为技术有限公司书面同意，任何单位和个人不得擅自摘抄、复制本手册内容的部分或全部，并不得以任何形式传播。

本手册中描述的产品中，可能包含华为技术有限公司及其可能存在的许可人享有版权的软件，除非获得相关权利人的许可，否则，任何人不得以任何形式对前述软件进行复制、分发、修改、摘录、反编译、反汇编、解密、反向工程、出租、转让、分许可以及其他侵犯软件版权的行为，但是适用法禁止此类限制的除外。

商标声明



HUAWEI、华为、 是华为技术有限公司的商标或者注册商标。

在本手册中以及本手册描述的产品中，出现的其他商标、产品名称、服务名称以及公司名称，由其各自的所有人拥有。

注意

本手册描述的产品及其附件的某些特性和功能，取决于当地网络的设计和性能，以及您安装的软件。某些特性和功能可能由于当地网络运营商或网络服务供应商不支持，或者由于当地网络的设置，或者您安装的软件不支持而无法实现。因此，本手册中的描述可能与您购买的产品或其附件并非完全一一对应。

华为技术有限公司保留随时修改本手册中任何信息的权利，无需进行任何提前通知且不承担任何责任。

无担保声明

本手册中的内容均“如是”提供，除非适用法要求，华为技术有限公司对本手册中的所有内容不提供任何明示或暗示的保证，包括但不限于适销性或者适用于某一特定目的的保证。

在法律允许的范围内，华为技术有限公司在任何情况下，都不对因使用本手册相关内容而产生的任何特殊的、附带的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉或预期节约的损失进行赔偿。

进出口管制

若需将此产品手册描述的产品（包含但不限于产品中的软件及技术数据等）出口、再出口或者进口，您应遵守适用的进出口管制法律法规



目录

1 目的.....	5
2 范围.....	6
3 总体概述.....	7
3.1 Linux 内核支持华为数据卡设备的驱动架构.....	7
4 Linux 内核驱动集成方案	9
4.1 Usb-storage 驱动集成流程.....	9
4.2 针对 2.6.29（包括 29）以下内核版本 Usb-storage 驱动的集成步骤.....	9
4.3 针对 2.6.29 以上内核版本 Usb-storage 驱动模块的集成步骤.....	13
4.4 USB 串口驱动集成的修改内容	16
4.5 内核配置项选择	18
4.5.1 Config 文件确认下述配置	18
4.5.2 驱动集成配置的操作步骤	19
5 附录.....	25
5.1 确认系统是否集成成功	25
5.2 若无法映射端口或无法查找对应端口形态需要提供下述 log.....	25



1 目的

本文档主要是针对华为模块设备基于 Linux 及 Android 系统的内核驱动集成开发活动进行相关的指导说明。主要面向基于 Linux 及 Android 系统的产品开发商的驱动开发人员。

2 范围

本文档适用于：

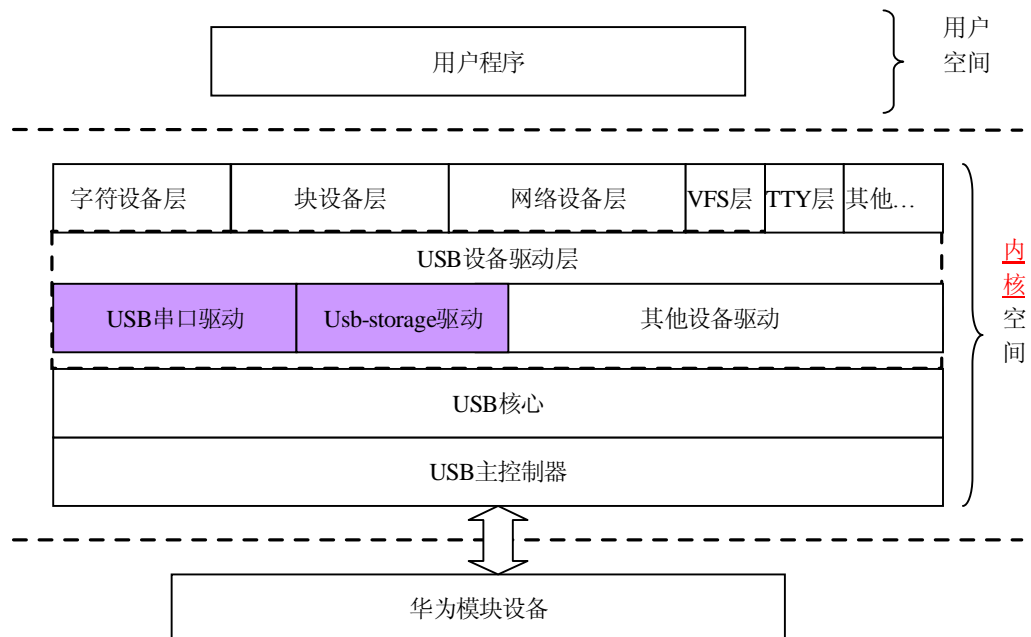
- I 华为 WCDMA/CDMA 制式的数据卡设备
- I 内核版本为 Linux 2.6.21 及以上的嵌入式 Linux 系统
- I 嵌入式 Linux 系统，网关(使用 Linux 内核)，Android 系统

3 总体概述

3.1 Linux 内核支持华为数据卡设备的驱动架构

华为数据卡设备和 Linux 系统主要通过 USB 接口进行数据通信。Linux 内核需要根据华为数据卡设备上报的 USB 设备接口加载 USB 驱动，USB 驱动正确加载后，数据卡才能正常工作。

Linux 系统中支持华为数据卡设备相关的 Linux 内核驱动架构，如下图所示：



如上图所示，在 Linux 系统中的 USB 驱动架构中，跟华为数据卡设备相关的驱动模块是 USB 串口驱动和 Usb-storage 驱动。其中：

I option 驱动模块

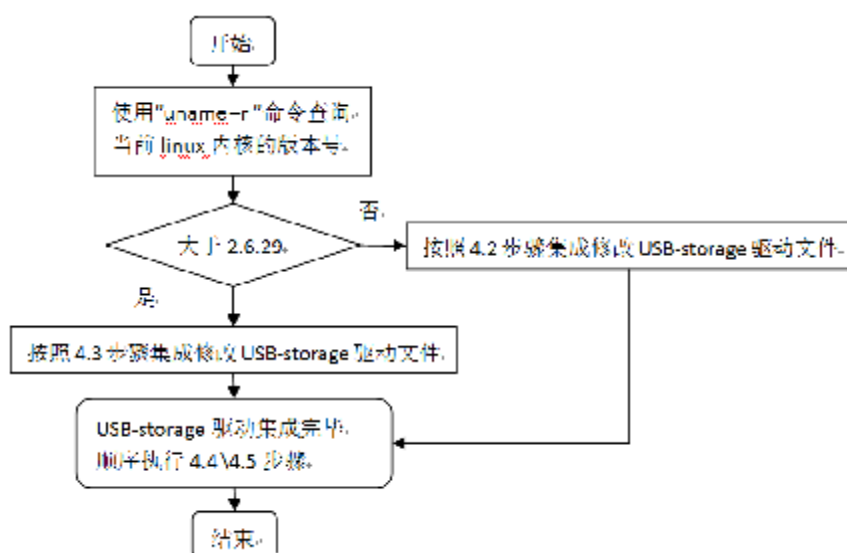
USB 的串口驱动，支持如 modem 端口，AT 端口等；该驱动模块的代码已经内置在 Linux 内核的源码中。

I usb-storage 驱动模块

USB 的光盘驱动，支持 USB 硬盘、磁盘、CD-ROM、读卡器。

4 Linux 内核驱动集成方案

4.1 Usb-storage 驱动集成流程



4.2 针对 2.6.29（包括 29）以下内核版本 Usb-storage 驱动的集成步骤

涉及的 Linux 内核源码文件有：

\\drivers\\usb\\storage\\usb.c

\\drivers\\usb\\storage\\unusual_devs.h

\\drivers\\usb\\storage\\initializers.c

\\drivers\\usb\\storage\\initializers.h

具体的修改步骤如下：

步骤 1 修改 usb.c 文件。

1. 在 “struct usb_device_id storage_usb_ids [] = {}” 语句前添加下述语句：

```
#define HW_UNUSUAL_DEV(id_vendor, cl, sc, pr, \
                        vendorName, productName, useProtocol, useTransport, \
                        initFunction, flags) \
```

```
{ \
    .match_flags = USB_DEVICE_ID_MATCH_INT_INFO | USB_DEVICE_ID_MATCH_VENDOR, \
    .idVendor    = (id_vendor), \
    .bInterfaceClass = (cl), \
    .bInterfaceSubClass = (sc), \
    .bInterfaceProtocol = (pr), \
    .driver_info = (flags)|(USB_US_TYPE_STOR<<24) }
```

在 “struct usb_device_id storage_usb_ids [] = {}” 语句中的 “#undef UNUSUAL_DEV” 语句后增加下述语句：

```
#undef HW_UNUSUAL_DEV
```

在 “static struct us_unusual_dev us_unusual_dev_list[] = {}” 语句前添加下述语句：

```
#define HW_UNUSUAL_DEV(idVendor, cl, sc, pr, \
                        vendor_name, product_name, use_protocol, use_transport, \
                        init_function, Flags) \
{ \
    .vendorName = vendor_name, \
    .productName = product_name, \
    .useProtocol = use_protocol, \
    .useTransport = use_transport, \
    .initFunction = init_function, \
}
```

2. 在 “static struct us_unusual_dev us_unusual_dev_list[] = {}” 语句中的 “#undef UNUSUAL_DEV” 语句后增加下述语句：

```
#undef HW_UNUSUAL_DEV
```

步骤 2 修改 unusual_devs.h 文件。

在原来的华为 vid(查找“12d1”关键字)声明的最后一个语句之后，添加特殊 USB 光盘设备声明语句(如果没有则可在任意位置添加)，如：

```
HW_UNUSUAL_DEV ( 0x12d1, 0x08, 0x06, 0x50,
                "HUAWEI",
                "HUAWEI MOBILE Mass Storage",
                US_SC_DEVICE, US_PR_DEVICE, usb_stor_huawei_init,
                0),
```

步骤 3 修改 initializers.c 文件。

1. 如果没有 `usb_stor_huawei_e220_init` 函数，请增加。如果已经存在，请直接执行下一步。

```
int usb_stor_huawei_e220_init(struct us_data *us)
{
    int result;

    result = usb_stor_control_msg(us, us->send_ctrl_pipe,
                                  USB_REQ_SET_FEATURE,
                                  USB_TYPE_STANDARD | USB_RECIP_DEVICE,
                                  0x01, 0x0, NULL, 0x0, 1000);

    US_DEBUGP("Huawei mode set result is %d\n", result);

    return 0;
}
```

2. 在 `usb_stor_huawei_e220_init` 函数后增加两个宏定义：

```
#define IS_HUAWEI_DONGLES 1
#define NOT_HUAWEI_DONGLES 0
```

3. 在文件中增加 “`usb_stor_huawei_dongles_pid`” 函数定义：

```
static int usb_stor_huawei_dongles_pid(struct us_data *us)
{
    int ret = NOT_HUAWEI_DONGLES;

    struct usb_interface_descriptor *idesc = NULL;

    idesc = &us->pusb_intf->cur_altsetting->desc;

    if (NULL != idesc ) {
        if ( (0x0000 == idesc->bInterfaceNumber)) {
            if ((0x1401 <= us->pusb_dev->descriptor.idProduct && 0x1600 >=
us->pusb_dev->descriptor.idProduct)
                || (0x1c02 <= us->pusb_dev->descriptor.idProduct &&
0x2202 >= us->pusb_dev->descriptor.idProduct)
                || (0x1001 == us->pusb_dev->descriptor.idProduct)
                || (0x1003 == us->pusb_dev->descriptor.idProduct)
                || (0x1004 == us->pusb_dev->descriptor.idProduct)) {
                if ((0x1501 <= us->pusb_dev->descriptor.idProduct) &&
(0x1504 >= us->pusb_dev->descriptor.idProduct)) {
                    ret = NOT_HUAWEI_DONGLES;
                } else {
                    ret = IS_HUAWEI_DONGLES;
                }
            }
        }
    }
```

```
        }  
    }  
}  
  
return ret;  
}
```

4. 接着在文件中增加 “usb_stor_huawei_scsi_init” 函数的定义:

```
int usb_stor_huawei_scsi_init(struct us_data *us)  
{  
    int result = 0;  
    int act_len = 0;  
    unsigned char cmd[32] = {0x55, 0x53, 0x42, 0x43, 0x00, 0x00, 0x00, 0x00,  
                             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x11,  
                             0x06, 0x30, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00,  
                             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };  
    result = usb_stor_bulk_transfer_buf (us, us->send_bulk_pipe, cmd, 31,  
    &act_len);  
    US_DEBUGP("usb_stor_bulk_transfer_buf performing result is %d, transfer  
the actual length=%d\n", result, act_len);  
    return result;  
}
```

5. 增加 usb_stor_huawei_init 函数

```
int usb_stor_huawei_init(struct us_data *us)  
{  
    int result = 0;  
    if(usb_stor_huawei_dongles_pid(us)){  
        if ((0x1446 <= us->pusb_dev->descriptor.idProduct )){  
            result = usb_stor_huawei_scsi_init(us);  
        }else{  
            result = usb_stor_huawei_e220_init(us);  
        }  
    }  
    return result;  
}
```

}

步骤 4 修改 initializers.h 文件。

接着在文件中增加 “usb_stor_huawei_init” 函数的声明：

```
int usb_stor_huawei_init(struct us_data *us);
```

4.3 针对 2.6.29 以上内核版本 Usb-storage 驱动模块的集成步骤

涉及的 Linux 内核源码文件有：

\drivers\usb\storage\usb.c

\drivers\usb\storage\usual-tables.c

\drivers\usb\storage\unusual_devs.h

\drivers\usb\storage\initializers.c

\drivers\usb\storage\initializers.h

具体的修改步骤如下：

步骤 1 修改 usb.c 文件。

1. 在 “static struct us_unusual_dev us_unusual_dev_list[] = {}” 语句前增加下述语句：

```
#define HW_UNUSUAL_DEV(idVendor, cl, sc, pr, \
                        vendor_name, product_name, use_protocol, use_transport, \
                        init_function, Flags) \
{\
    .vendorName = vendor_name, \
    .productName = product_name, \
    .useProtocol = use_protocol, \
    .useTransport = use_transport, \
    .initFunction = init_function, \
}
```

2. 在 “#undef COMPLIANT_DEV” 语句后增加下述语句：

```
#undef HW_UNUSUAL_DEV
```

步骤 2 修改 usual-tables.c 文件。

1. 在 “struct usb_device_id storage_usb_ids [] = {}” 语句前增加下述语句：

```
#define HW_UNUSUAL_DEV(id_vendor, cl, sc, pr, \
                        vendorName, productName, useProtocol, useTransport, \
                        initFunction, flags) \
{\
    .match_flags = USB_DEVICE_ID_MATCH_INT_INFO | USB_DEVICE_ID_MATCH_VENDOR, \
```

```
.idVendor    = (id_vendor), \
.bInterfaceClass = (cl), \
.bInterfaceSubClass = (sc), \
.bInterfaceProtocol = (pr), \
.driver_info = (flags)|(USB_US_TYPE_STOR<<24) }
```

2. 在 “#undef COMPLIANT_DEV” 语句后增加下述语句:

```
#undef HW_UNUSUAL_DEV
```

步骤 3 修改 unusual_devs.h 文件。

在原来的华为(查找“12d1”关键字)声明的最后一个语句之后, 添加特殊 USB 光盘设备声明语句(如果没有则可在任意位置添加), 如:

```
HW_UNUSUAL_DEV ( 0x12d1, 0x08, 0x06, 0x50,
    "HUAWEI",
    "HUAWEI MOBILE Mass Storage",
    US_SC_DEVICE, US_PR_DEVICE, usb_stor_huawei_init,
    0),
```

(Linux-2.6.37 及以上内核版本,上述红色字体应该修改为 USB_SC_DEVICE,USB_PR_DEVICE)

步骤 4 修改 initializers.c 文件。

1. 在 usb_stor_huawei_e220_init 函数后增加下述两宏定义:

```
#define IS_HUAWEI_DONGLES 1
#define NOT_HUAWEI_DONGLES 0
```

2. 在文件中增加 “usb_stor_huawei_dongles_pid” 函数定义:

```
static int usb_stor_huawei_dongles_pid(struct us_data *us)
{
    int ret = NOT_HUAWEI_DONGLES;

    struct usb_interface_descriptor *idesc = NULL;

    idesc = &us->pusb_intf->cur_altsetting->desc;

    if (NULL != idesc ) {
        if ( (0x0000 == idesc->bInterfaceNumber)) {
            if ((0x1401 <= us->pusb_dev->descriptor.idProduct && 0x1600 >=
us->pusb_dev->descriptor.idProduct)

                || (0x1c02 <= us->pusb_dev->descriptor.idProduct &&
0x2202 >= us->pusb_dev->descriptor.idProduct)

                || (0x1001 == us->pusb_dev->descriptor.idProduct)

                || (0x1003 == us->pusb_dev->descriptor.idProduct)

                || (0x1004 == us->pusb_dev->descriptor.idProduct)) {
```

```

        if ((0x1501 <= us->pusb_dev->descriptor.idProduct) &&
(0x1504 >= us->pusb_dev->descriptor.idProduct)) {

            ret = NOT_HUAWEI_DONGLES;

        } else {

            ret = IS_HUAWEI_DONGLES;

        }

    }

}

return ret;
}

```

3. 接着在文件中增加“usb_stor_huawei_scsi_init”函数的定义：

```

int usb_stor_huawei_scsi_init(struct us_data *us)
{
    int result = 0;

    int act_len = 0;

    unsigned char cmd[32] = {0x55, 0x53, 0x42, 0x43, 0x00, 0x00, 0x00, 0x00,
                                0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x11,
                                0x06, 0x30, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00,
                                0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

    result = usb_stor_bulk_transfer_buf (us, us->send_bulk_pipe, cmd, 31,
&act_len);

    US_DEBUGP("usb_stor_bulk_transfer_buf performing result is %d, transfer
the actual length=%d\n", result, act_len);

    return result;
}

```

4. 增加 usb_stor_huawei_init 函数：

```

int usb_stor_huawei_init(struct us_data *us)
{
    int result = 0;

    if(usb_stor_huawei_dongles_pid(us)){

        if ((0x1446 <= us->pusb_dev->descriptor.idProduct )){

            result = usb_stor_huawei_scsi_init(us);

        }else{

            result = usb_stor_huawei_e220_init(us);

        }

    }

}

```

```
    }  
}  
  
return result;  
}
```

步骤 5 修改 `initializers.h` 文件。.

接着在文件中增加 “`usb_stor_huawei_init`” 函数的声明：

```
int usb_stor_huawei_init(struct us_data *us);
```

4.4 USB 串口驱动集成的修改内容

涉及到的 Linux 内核源码文件是：

`drivers/usb/serial/option.c`

增加了对新 PID 的支持，修改部分如下：

在“`static const struct usb_device_id option_ids[]`”前增加下述定义：

```
#define HW_USB_DEVICE_AND_INTERFACE_INFO(vend, cl, sc, pr) \  
    .match_flags = USB_DEVICE_ID_MATCH_INT_INFO \  
        | USB_DEVICE_ID_MATCH_VENDOR, \  
    .idVendor = (vend), \  
    .bInterfaceClass = (cl), \  
    .bInterfaceSubClass = (sc), \  
    .bInterfaceProtocol = (pr)
```

在 “`static const struct usb_device_id option_ids[]`” id 列表中增加如下语句：

```
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0x02, 0x02, 0xff) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0xff, 0xff) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x01) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x02) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x03) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x04) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x05) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x06) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x0A) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x0B) },  
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x0D) },
```

```
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x0E) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x0F) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x10) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x12) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x13) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x14) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x15) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x31) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x32) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x33) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x34) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x35) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x36) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x3A) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x3B) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x3D) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x3E) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x01, 0x3F) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x01) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x02) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x03) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x04) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x05) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x06) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x0A) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x0B) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x0D) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x0E) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x0F) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x10) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x12) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x13) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x14) },
```



```
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x15) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x31) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x32) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x33) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x34) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x35) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x36) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x3A) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x3B) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x3D) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x3E) },
{ HW_USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, 0xff, 0x02, 0x3F) },
```

4.5 内核配置项选择

修改 Linux 内核的编译配置(在 **kernel** 根目录下的 **.config** 文件中)，确保下面的的配置项已经被选定(enabled)。具体的配置操作请参考 4.5.2 节。

4.5.1 Config 文件确认下述配置

1. USB 串口驱动相关的配置项：
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_OPTION=y
2. USB 网口驱动相关的配置项：
CONFIG_MII=y
CONFIG_USBNET=y
CONFIG_USB_NET_CDCETHER=y
3. PPP 拨号的相关配置项：
CONFIG_PPP=y
CONFIG_PPP_MULTILINK=y
CONFIG_PPP_FILTER=y
CONFIG_PPP_ASYNC=y
CONFIG_PPP_SYNC_TTY=y
CONFIG_PPP_DEFLATE=y
CONFIG_PPP_BSDCOMP=y

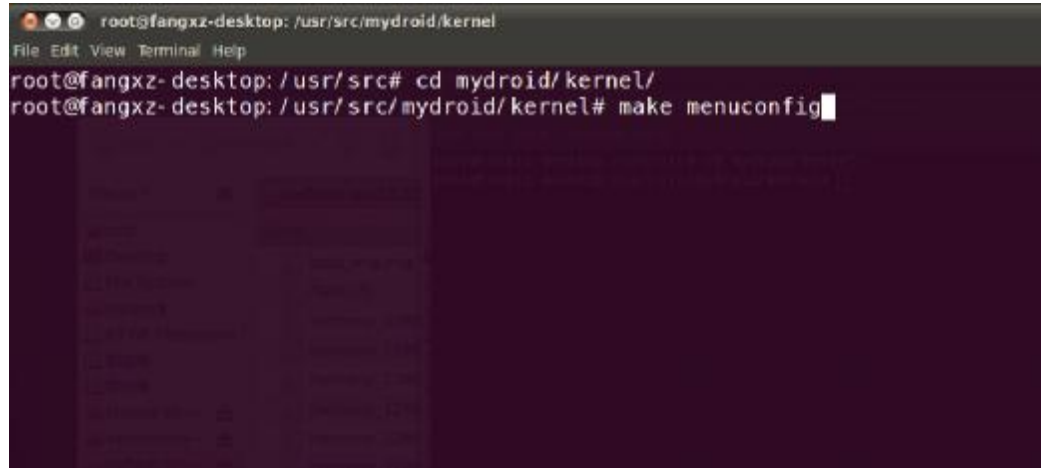


说明

在 linux-2.6.29 以下的内核不需要编译 CONFIG_USB_LIBUSUAL。参照 4.5.2 第 2 步第 1 小节。

4.5.2 驱动集成配置的操作步骤

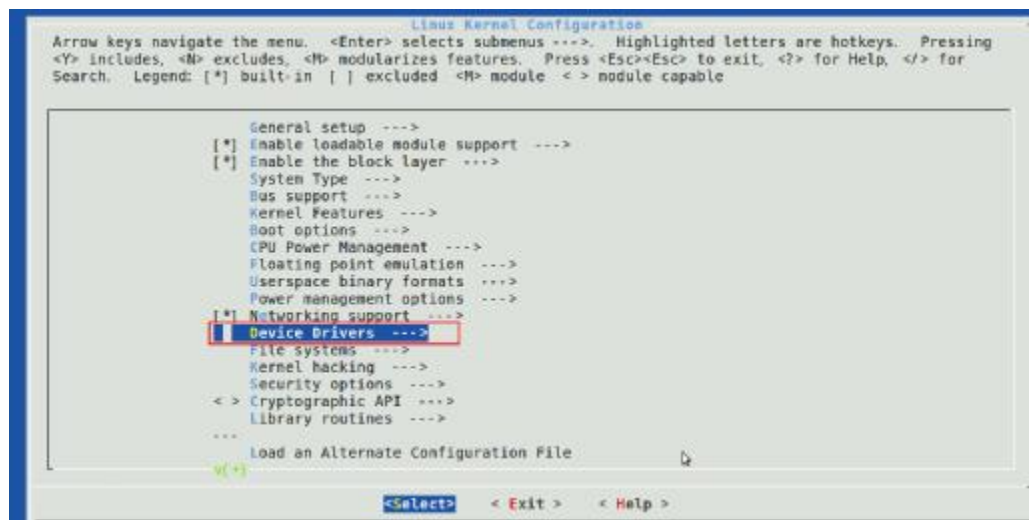
步骤 1 打开 Terminal 工具，进入 kernel 目录（假定 kernel 在 /usr/src/mydroid/ 目录下，即 cd /usr/src/mydroid/kernel），然后执行 make <configuration> 命令（在本文中，假定使用标准的 make menuconfig）。

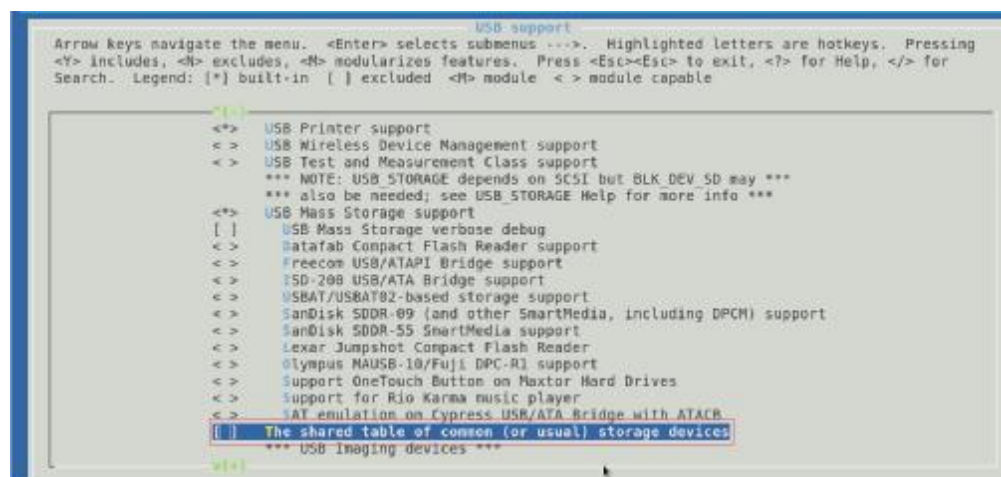
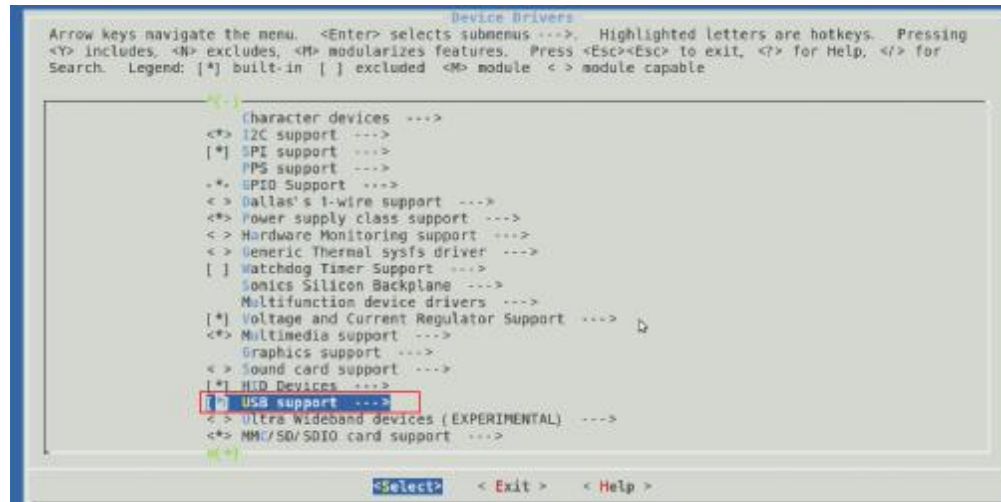


步骤 2 选择相关的配置项

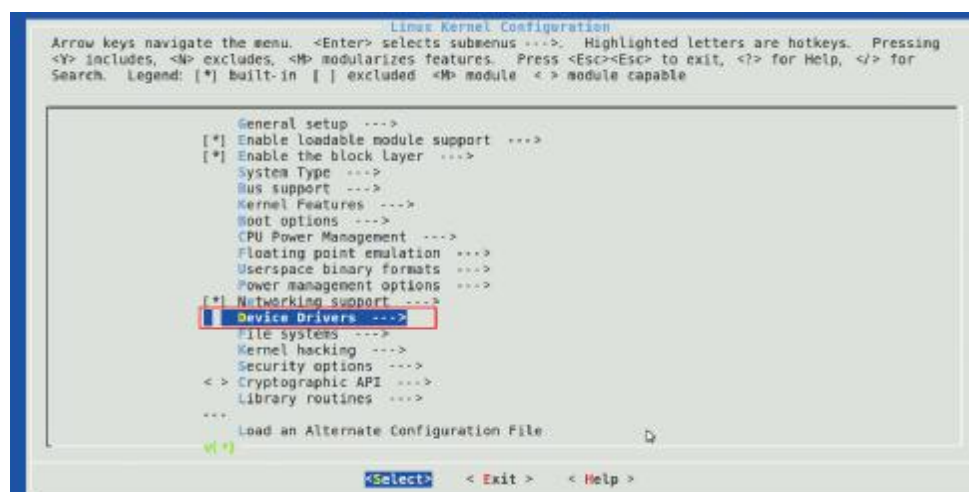
1. 不编译 CONFIG_USB_USUAL 配置选项

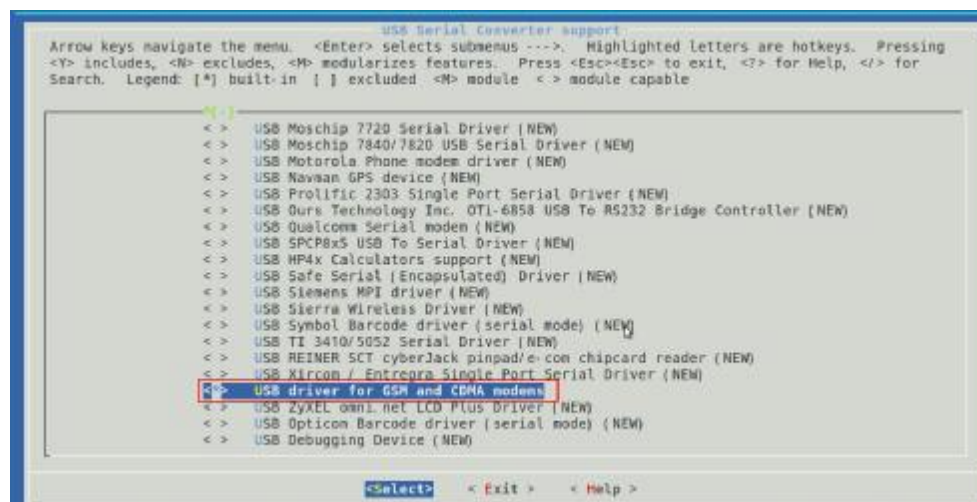
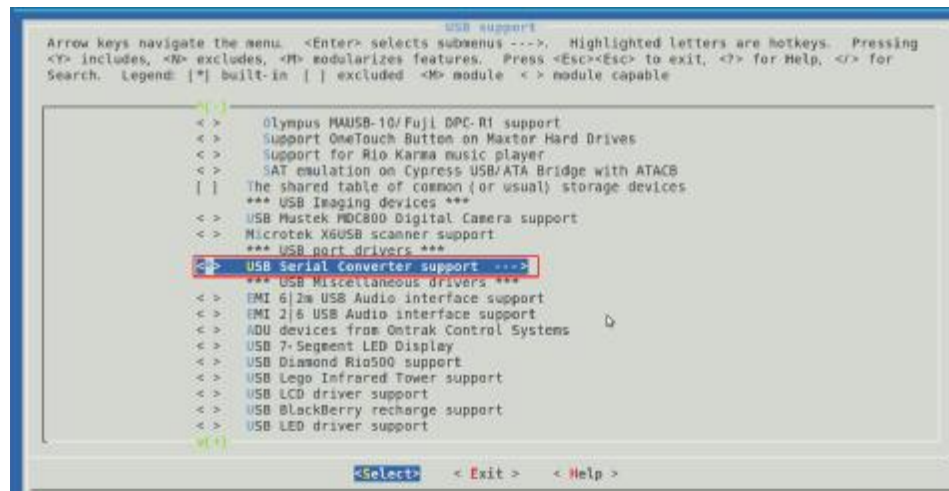
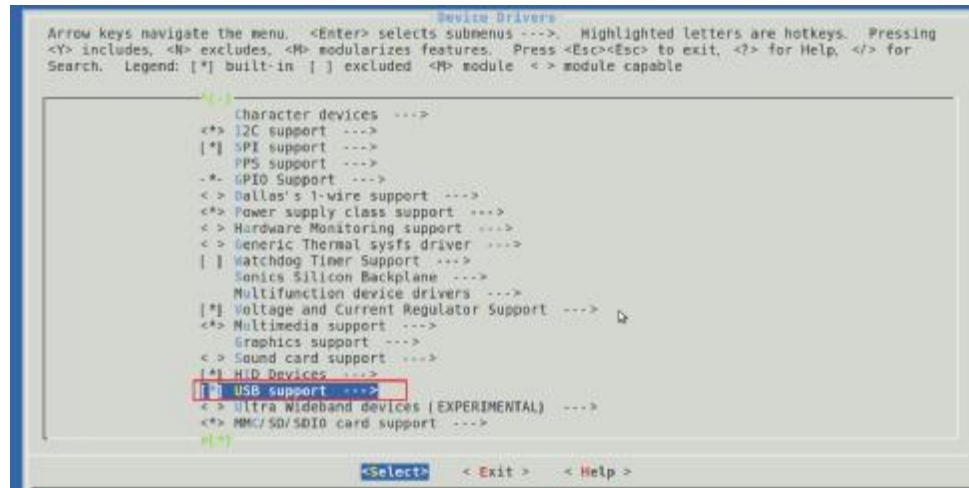
即将此项 “[] The shared table of common (or usual) storage devices” 配置为空即可，依图操作即可：



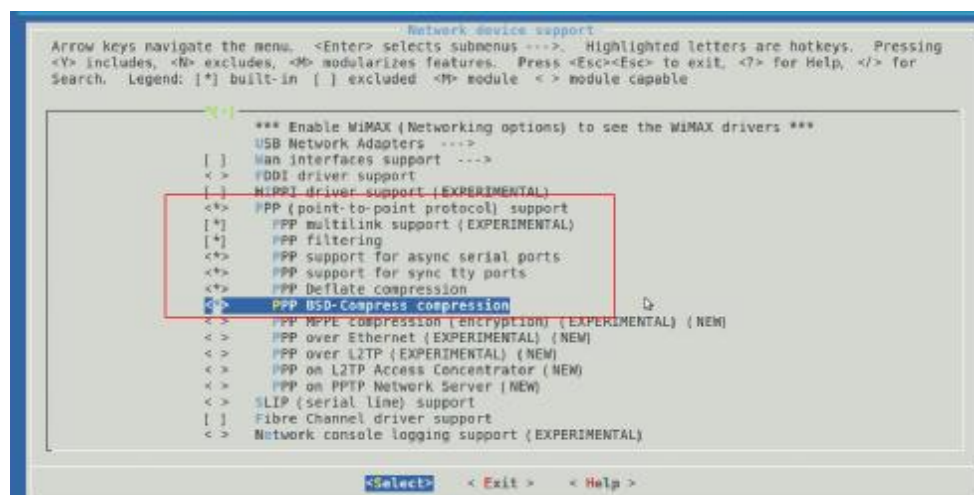
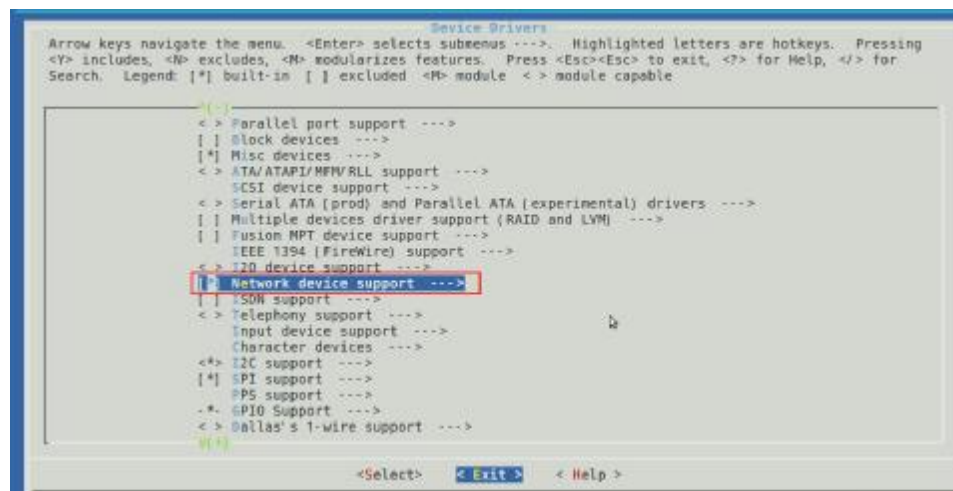
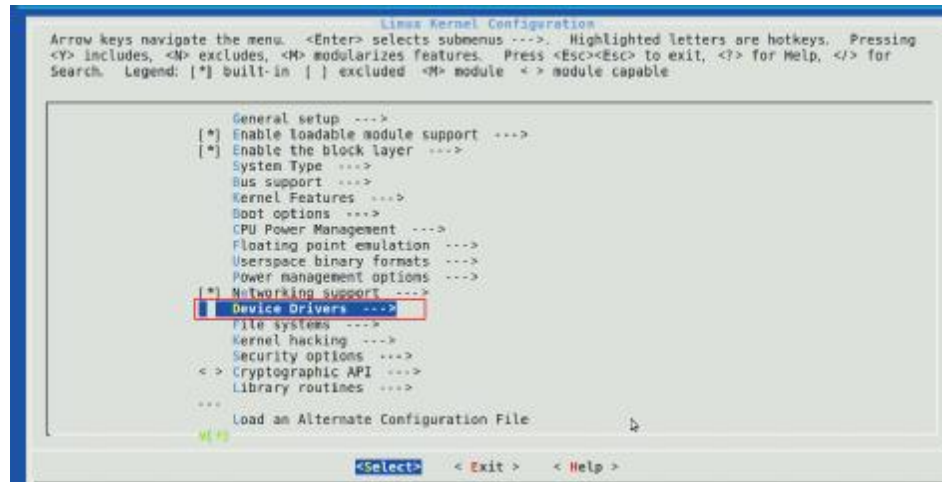


2. USB 串口驱动相关的配置项

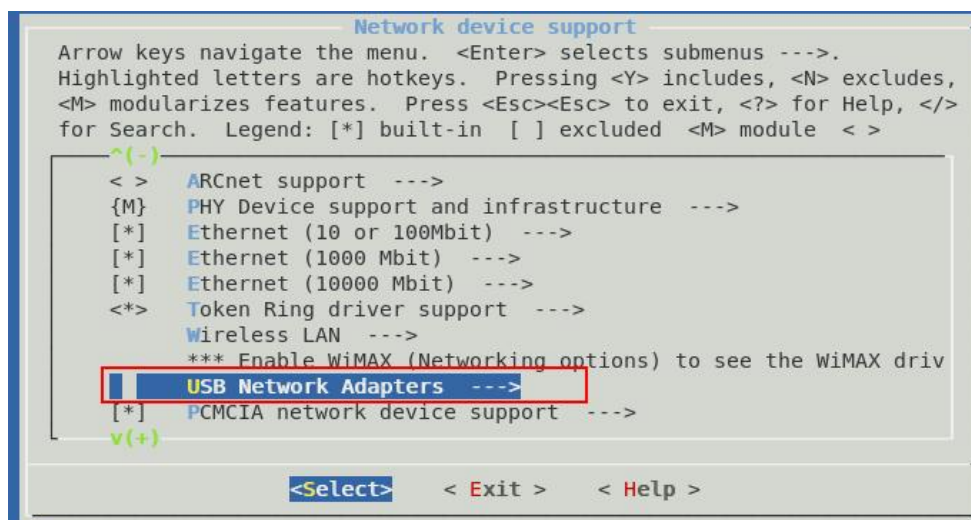
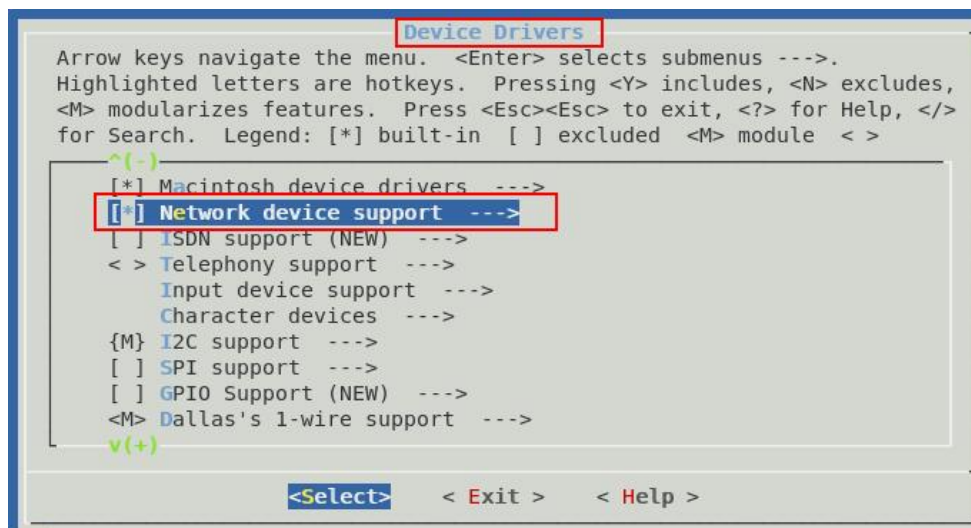




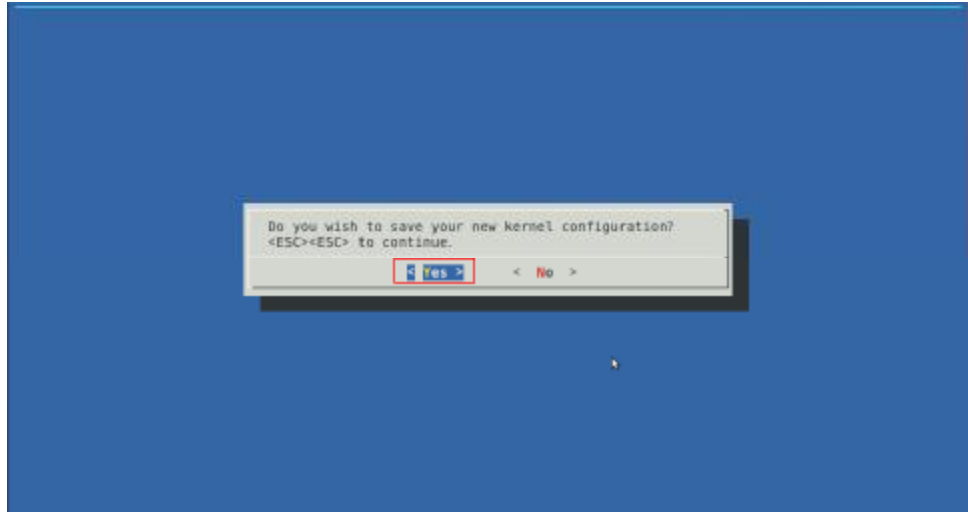
3. PPP 拨号的相关配置项



4. USB 网口驱动相关配置项



步骤 3 如上操作选完所需选项后，通过选择<Exit>按钮，逐层退出各个配置界面。最后在保存配置界面中，选择“Yes”选项并退出。



步骤 4 完成配置后，即可运行 `make` 命令，开始编译修改后的内核。

5 附录

5.1 确认系统是否集成成功

执行“dmesg”命令，查看集成是否成功,假如 log 信息中存在如下(或相似)的信息，则说明集成成功(idProduct 因产品的不同而不同, ttyUSB* 的个数也跟产品相关):

```
[276584.798530] usb 2-1.4: new high speed USB device using ehci-hcd and address 8
[276584.883888] usb 2-1.4: New USB device found, idVendor=12d1, idProduct=1484
[276584.883812] usb 2-1.4: New USB device strings: Mfr=3, Product=2, SerialNumber=0
[276584.883816] usb 2-1.4: Product: HUAWEI MOBILE WCDMA CM770W
[276584.883819] usb 2-1.4: Manufacturer: HUAWEI Technology
[276584.978546] USB Serial support registered for GSM modem (1-port)
[276584.978717] option 2-1.4:1.0: GSM modem (1-port) converter detected
[276584.978970] usb 2-1.4: GSM modem (1-port) converter now attached to ttyUSB0
[276584.978987] option 2-1.4:1.1: GSM modem (1-port) converter detected
[276584.979005] usb 2-1.4: GSM modem (1-port) converter now attached to ttyUSB1
[276584.979100] option 2-1.4:1.2: GSM modem (1-port) converter detected
[276584.979162] usb 2-1.4: GSM modem (1-port) converter now attached to ttyUSB2
[276584.979175] option 2-1.4:1.3: GSM modem (1-port) converter detected
[276584.979234] usb 2-1.4: GSM modem (1-port) converter now attached to ttyUSB3
[276584.979248] option 2-1.4:1.4: GSM modem (1-port) converter detected
[276584.979312] usb 2-1.4: GSM modem (1-port) converter now attached to ttyUSB4
```

执行“ls /dev/ttyUSB*”命令查询端口是否映射成功:

```
# ls /dev/ttyU*
/dev/ttyUSB0
/dev/ttyUSB1
/dev/ttyUSB2
/dev/ttyUSB3
/dev/ttyUSB4
```

5.2 若无法映射端口或无法查找对应端口形态需要提供下述 log

- I 打开 terminal,执行“dmesg”命令并保存输出结果于 dmesg.txt 文档。
- I 执行“ls -l /sys/bus/usb/devices/”命令保存输出结果于 logcat.txt 文档。
- I 执行“ls -l /sys/bus/usb/drivers/”命令保存输出结果于 logcat.txt 文档，确保此目录下有 option 选项，如果没有请参照 4.5.2 节检查一次确认修改正确。
- I 执行“ls -l /sys/bus/usb/drivers/option/**/”命令保存输出结果于 logcat.txt 文档
- I 执行“cat /sys/bus/usb/drivers/option/**/blInterface*”命令保存输出结果于 logcat.txt 文档

- I 红色部分为类似于“1-1:1.0”“1-1:1.1”“1-1:1.*”这种结构的阿拉伯字数。
- I 如果能执行“cat /proc/bus/usb/devices”此命令，请将输出结果保存于 logcat.txt 文档(如果不能请忽略此命令)