# Linux 3.8 内核以下模块驱动加载

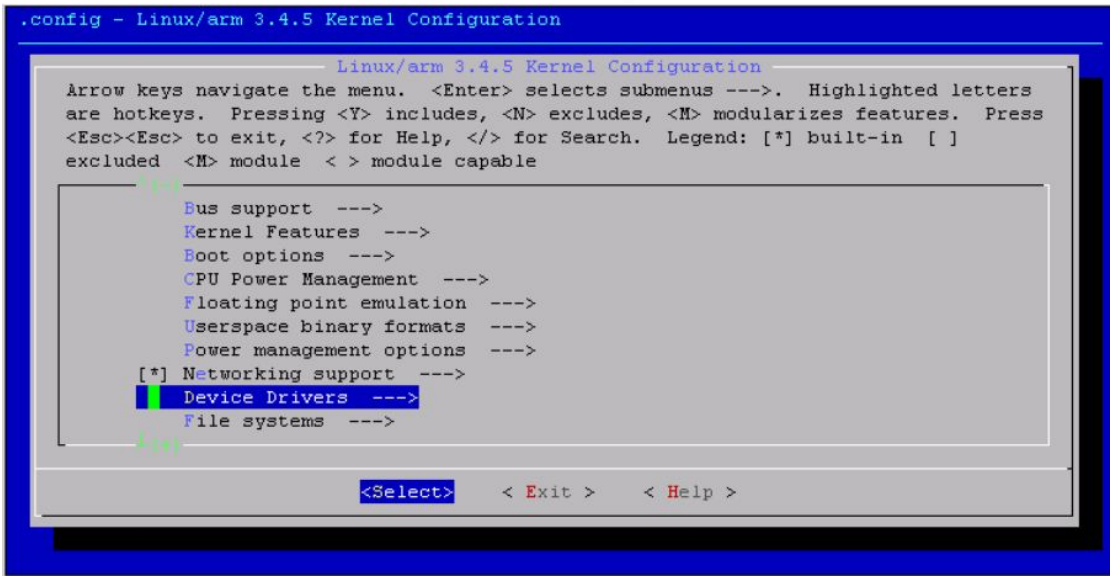
## 1、USB 驱动安装与加载

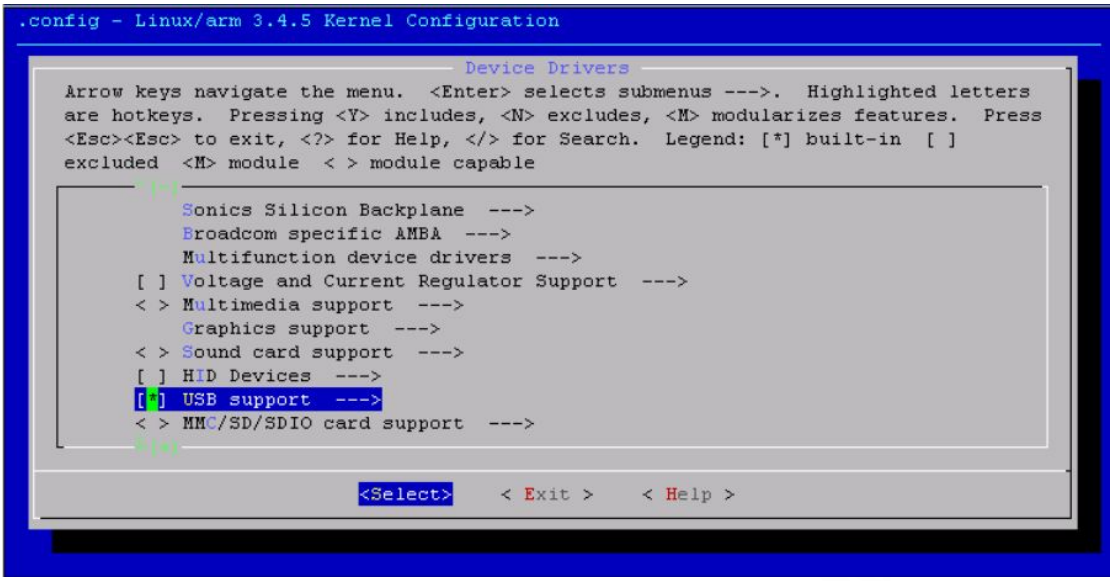
### 1.1 USB 信息

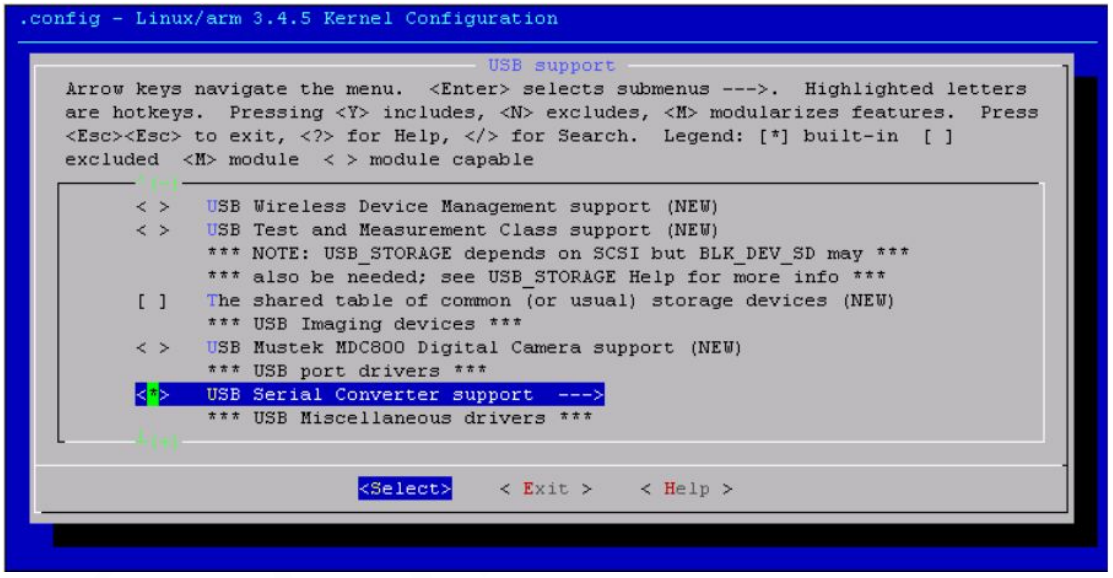
| VID | PID | 接口号 | 注释 |
|---|---|---|---|
| 0x19D2 | 0x0199 | 0 | ZTE CMCC AT Interface |
| | | 1 | 以太网卡 |
| | | 2 | ZTE CMCC Modem Interface |
| | | 3 | ZTE CMCC Log Interface |

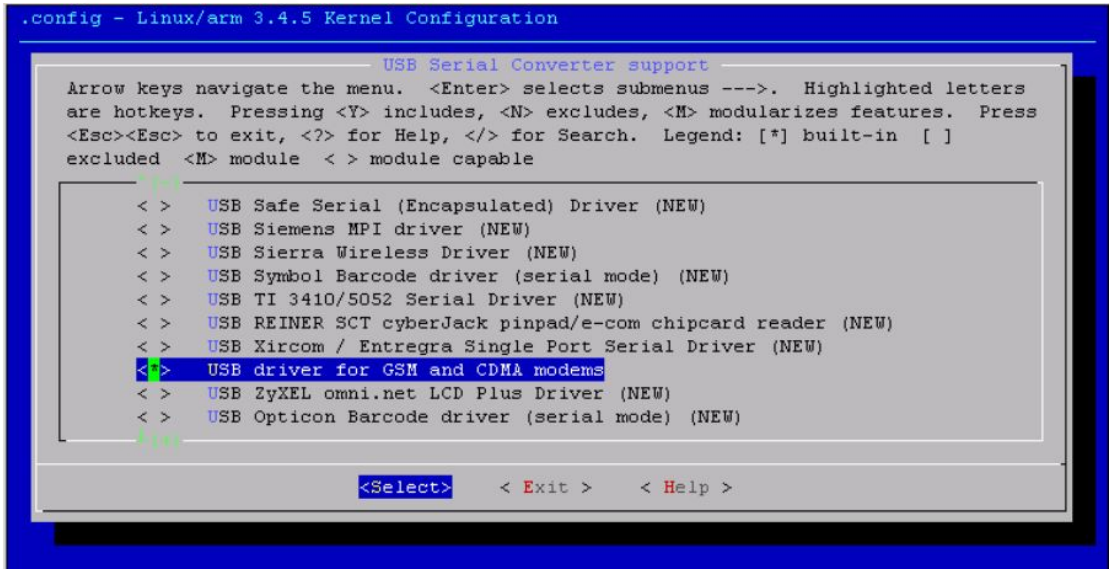
### 1.2 USB 串口驱动的安装

USB 接口中的 0 、2 、3 三个接口需要安装 USB 转串口驱动。需要编译内核的 drivers/usb/serial 模块，编译 bus.c generic.c option.c usb-serial.c usb_wwan.c 四个文件，其中 usb_wwan.c 在较早的 Linux 版本中没有。 通过 make menuconfig 配置内核编译该模块的方法如下：




.config - Linux/arm 3.4.5 Kernel Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
Sonics Silicon Backplane --->
Broadcom specific AMBA --->
Multifunction device drivers --->
[ ] Voltage and Current Regulator Support --->
< > Multimedia support --->
Graphics support --->
< > Sound card support --->
[ ] HID Devices --->
[*] USB support --->
< > MMC/SD/SDIO card support --->
<Select> < Exit > < Help >


.config - Linux/arm 3.4.5 Kernel Configuration
USB support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
< > USB Wireless Device Management support (NEW)
< > USB Test and Measurement Class support (NEW)
*** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
*** also be needed; see USB_STORAGE Help for more info ***
[ ] The shared table of common (or usual) storage devices (NEW)
*** USB Imaging devices ***
< > USB Mustek MDC800 Digital Camera support (NEW)
*** USB port drivers ***
<*> USB Serial Converter support --->
*** USB Miscellaneous drivers ***
<Select> < Exit > < Help >


.config - Linux/arm 3.4.5 Kernel Configuration
USB Serial Converter support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
< > USB Safe Serial (Encapsulated) Driver (NEW)
< > USB Siemens MPI driver (NEW)
< > USB Sierra Wireless Driver (NEW)
< > USB Symbol Barcode driver (serial mode) (NEW)
< > USB TI 3410/5052 Serial Driver (NEW)
< > USB REINER SCT cyberJack pinpad/e-com chipcard reader (NEW)
< > USB Xircom / Entregra Single Port Serial Driver (NEW)
<*> USB driver for GSM and CDMA modems
< > USB ZyXEL omni.net LCD Plus Driver (NEW)
< > USB Opticon Barcode driver (serial mode) (NEW)
<Select> < Exit > < Help >

在配置完内核后，需要修改文件 drivers/usb/serial/option.c，增加对模块 USB 的支持。首先是在 option.c 文件中找到全局变量 option_ids，将 USB 的信息添加到 option_ids 数组中，在数组中增加成员值如下：

```
{ USB_DEVICE_AND_INTERFACE_INFO(ZTE_VENDOR_ID,ZTE_PRODUCT_ZX297510MDM, 0xff, 0xff, 0xff) },/* ZX297510 products */
```

其中宏 ZTE_VENDOR_ID 和 ZTE_PRODUCT_ZX297510MDM 定义如下：

```
#define ZTE_VENDOR_ID              0x19d2
#define ZTE_PRODUCT_ZX297510MDM    0x0199
```

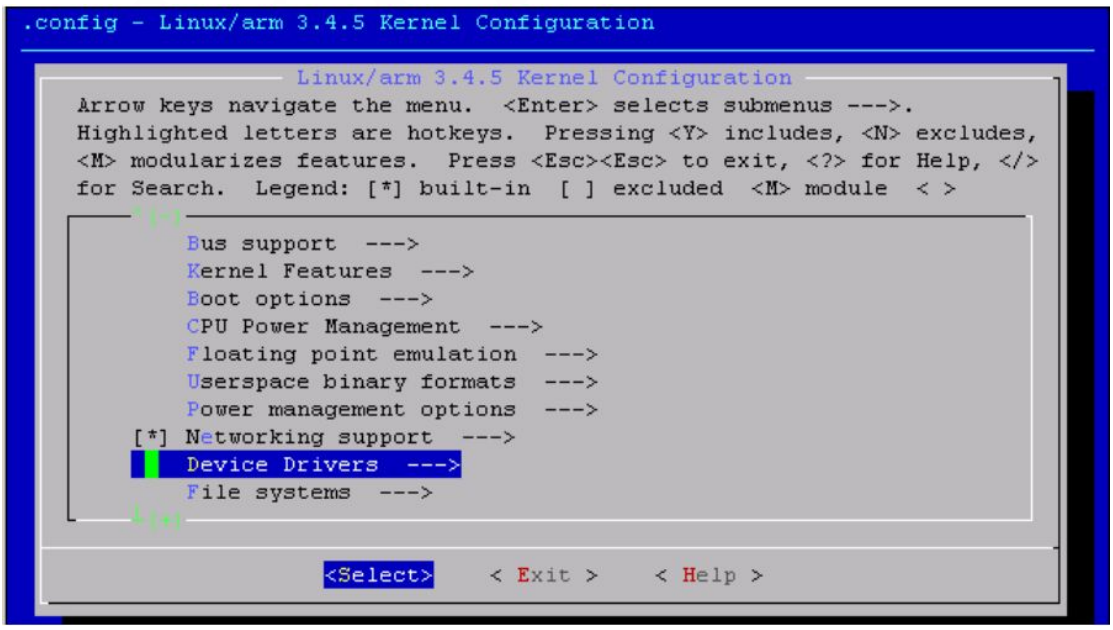
然后修改 option_probe 函数将接口 1 从 USB 串口驱动中过滤过来，如下：

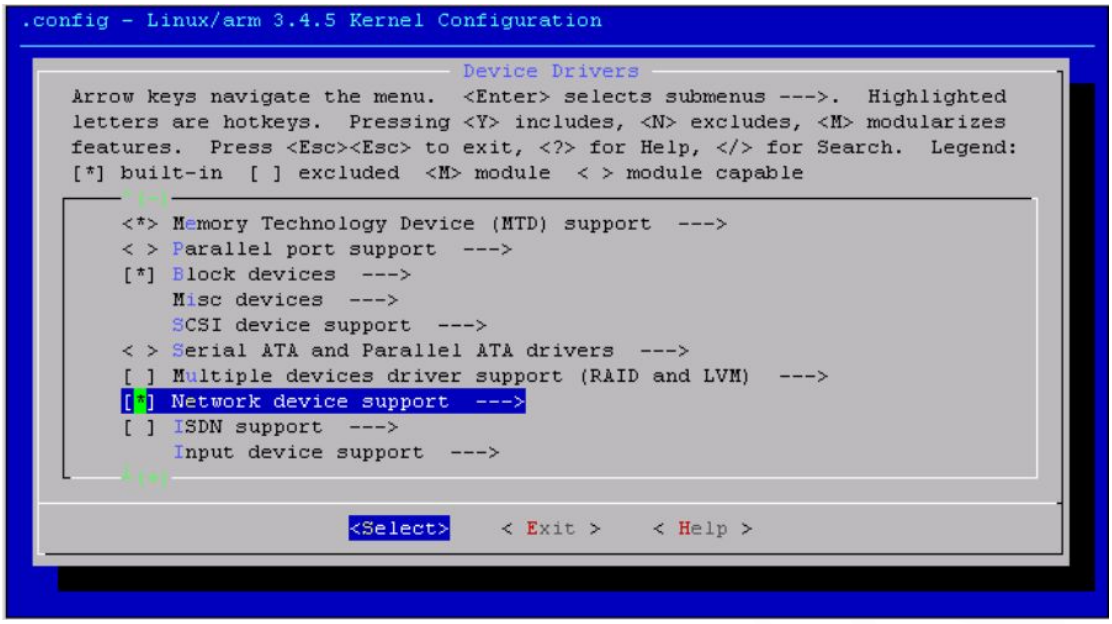
```
if (serial->dev->descriptor.idVendor ==   ZTE_VENDOR_ID &&
serial->dev->descriptor.idProduct == ZTE_PRODUCT_ZX297510MDM &&
serial->interface->cur_altsetting->desc.bInterfaceNumber == 0x1)
{
   printk("serial option_probe 0x%X - 0x%X - %d\n",
   serial->dev->descriptor.idVendor,
   serial->dev->descriptor.idProduct,
   serial->interface->cur_altsetting->desc.bInterfaceNumber);
   return -ENODEV;
}
```

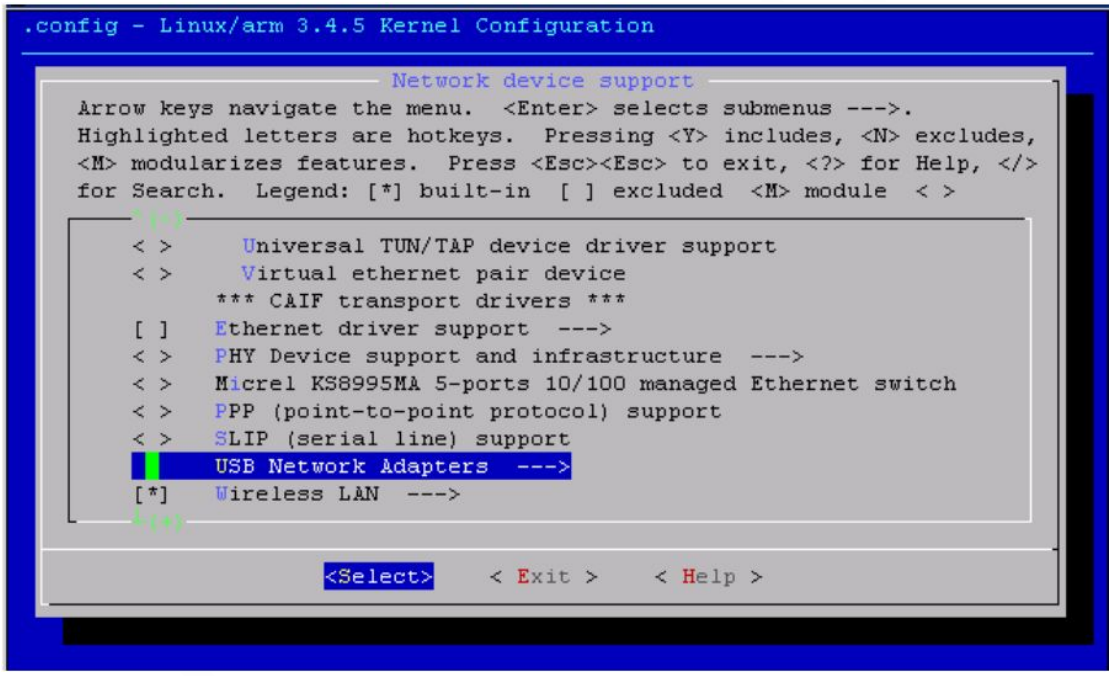
## 1.3 网卡驱动安装

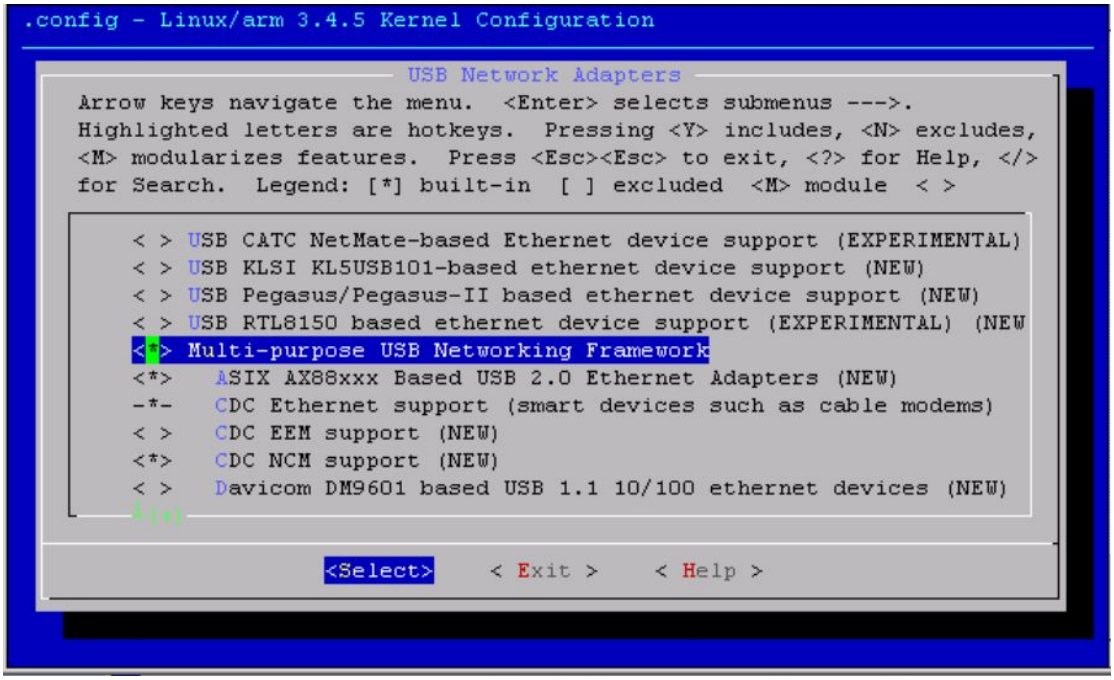
USB 接口中的接口 1 为网络数据接口。如果设备枚举的是标准的 CDC 类网络设备（如 ECM）驱动的加载不在这里描述。

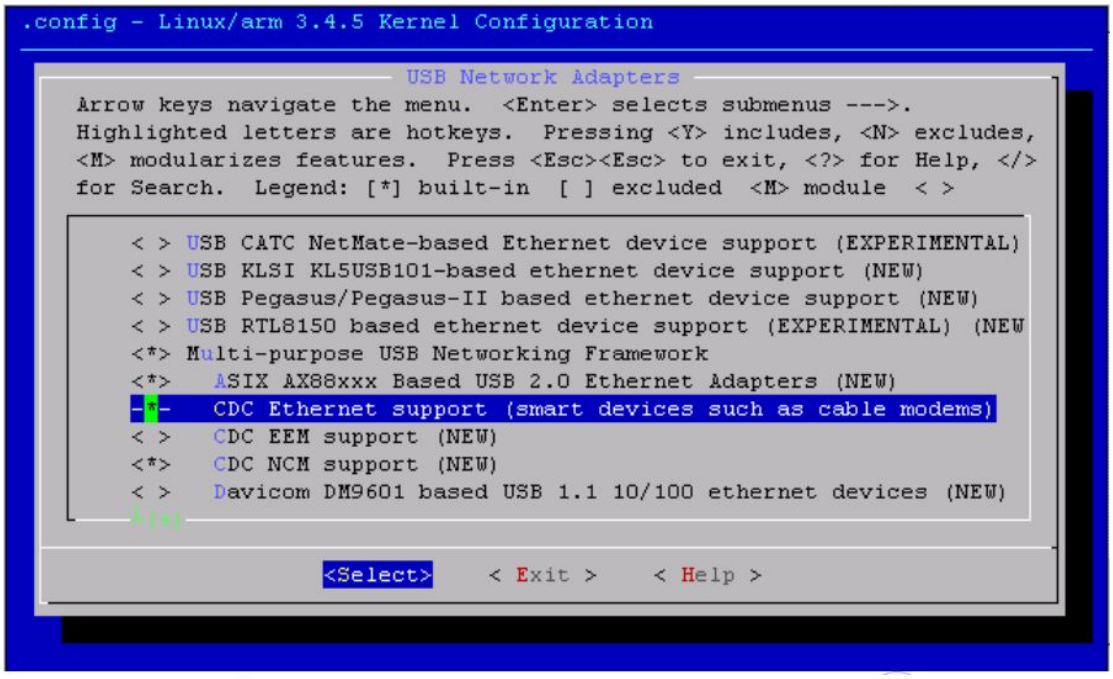
首先需要配置将文件 drivers/net/usb 下的 cdc_ether.c 和 usbnet.c 编译到内核， 通过 make menuconfig 配置内核编译该模块




.config - Linux/arm 3.4.5 Kernel Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> Memory Technology Device (MTD) support --->
< > Parallel port support --->
[*] Block devices --->
Misc devices --->
SCSI device support --->
< > Serial ATA and Parallel ATA drivers --->
[ ] Multiple devices driver support (RAID and LVM) --->
[*] Network device support --->
[ ] ISDN support --->
Input device support --->
<Select>
< Exit >
< Help >


.config - Linux/arm 3.4.5 Kernel Configuration
Network device support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < >
< > Universal TUN/TAP device driver support
< > Virtual ethernet pair device
*** CAIF transport drivers ***
[ ] Ethernet driver support --->
< > PHY Device support and infrastructure --->
< > Micrel KS8995MA 5-ports 10/100 managed Ethernet switch
< > PPP (point-to-point protocol) support
< > SLIP (serial line) support
USB Network Adapters --->
[*] Wireless LAN --->
<Select>
< Exit >
< Help >




内核配置完成后，需要修改内核 drivers/net/usb/cdc_ether.c 文件来支持网卡的驱动。首先在 cdc_ether.c 文件的 products 全局数组中增加对 USB 设备的支持，增加成员如下：

```
{
/*  ZTE ZX297510 MDM ether*/
USB_DEVICE_AND_INTERFACE_INFO(0x19D2, 0x0199, 0xFF, 0xFF, 0xFF),
.driver_info = (unsigned long) &zte_zx297510_mdm_ether_info_0199,
},
```

其中变量 zte_zx297510_mdm_ether_info_0199 定义为:

```
static const struct driver_info    zte_zx297510_mdm_ether_info_0199 = {
.description =     "ZTE Ethernet Device",
.flags =              FLAG_ETHER,
.bind =               cdc_bind,
.unbind =             usbnet_cdc_unbind,
.status =             cdc_status,
.data =               (unsigned long) &zte_zx297510_mdm_ether_iface_info_0199,
};
```

注意：Linux2.6 内核中的 cdc_bind 和 cdc_status 函数在 Linux3.0 与 Linux3.4 等高版本内核中为 usbnet_cdc_bind 和 usbnet_cdc_status。

变量 zte_zx297510_mdm_ether_iface_info_0199 定义为：

```
static const struct cdc_iface_info          zte_zx297510_mdm_ether_iface_info_0199 = {
      .iface_number = 1
};
```

结构体 cdc_iface_info 定义为：

```
struct cdc_iface_info {
__u8 iface_number;
};
```

然后修改驱动的 probe 函数。将 cdc_driver 的 probe 成员由原来的 usbnet_probe 修改成新增的 cdc_probe 函数。cdc_probe 先是对接口进行检查，如果是结构 cdc_iface_info 中描述的接口，则继续调用 usbnet_probe 函数，否则返回-ENODEV。cdc_probe 函数实现如下：

```
static int cdc_probe(struct usb_interface *intf, const struct usb_device_id *prod)
{
      struct driver_info  *info;
      int                 status;
      struct usb_device       *xdev;

      xdev = interface_to_usbdev (intf);
      if(xdev->descriptor.idVendor == 0x19D2 && xdev->descriptor.idProduct == 0x0199)
      {
            info = (struct driver_info *) prod->driver_info;

            if (info)
                  dev_dbg(&intf->dev, "%s: Probe\n", info->description);

            if (info && info->data)
            {
                  __u8                  iface_num;
                  struct cdc_iface_info   *iface_info;
```

```
                iface_num = intf->cur_altsetting->desc.bInterfaceNumber;
                iface_info = (struct cdc_iface_info *) info->data;

                dev_dbg (&intf->dev, "%s: trying iface %d\n",
                    info->description, iface_num);

                if (iface_info->iface_number != iface_num)
                    return -ENODEV;

                dev_info(&intf->dev, "%s: claiming interface %d\n",
                    info->description, iface_num);
            }
        }

        status = usbnet_probe(intf, prod);
        printk("cdc_probe   usbnet_probe return %d\n", status);
        if (status < 0)
            return status;

        return 0;
}
```

最后修改 cdc_bind 或 usbnet_cdc_bind 函数。由于这里的网络接口是非标准的设备接口，因此 usbnet_generic_cdc_bind 函数必定失败，因此在 usbnet_generic_cdc_bind 返回失败后需要继续调用 usbnet_get_endpoints 函数。

```
static int cdc_bind(struct usbnet *dev, struct usb_interface *intf)
{
int             status;
struct cdc_state       *info = (void *) &dev->data;
status = usbnet_generic_cdc_bind(dev, intf);
if (status < 0) {
status = usbnet_get_endpoints(dev, intf);
if (status < 0)
return status;
}
if (info->ether) {
printk("\r\n %s,[%d]",__FUNCTION__,__LINE__);
status = usbnet_get_ethernet_addr(dev,info->ether->iMACAddress);
if (status < 0)
goto error;
}
/* FIXME cdc-ether has some multicast code too, though it complains
* in routine cases.      info->ether describes the multicast support.
* Implement that here, manipulating the cdc filter as needed.
```

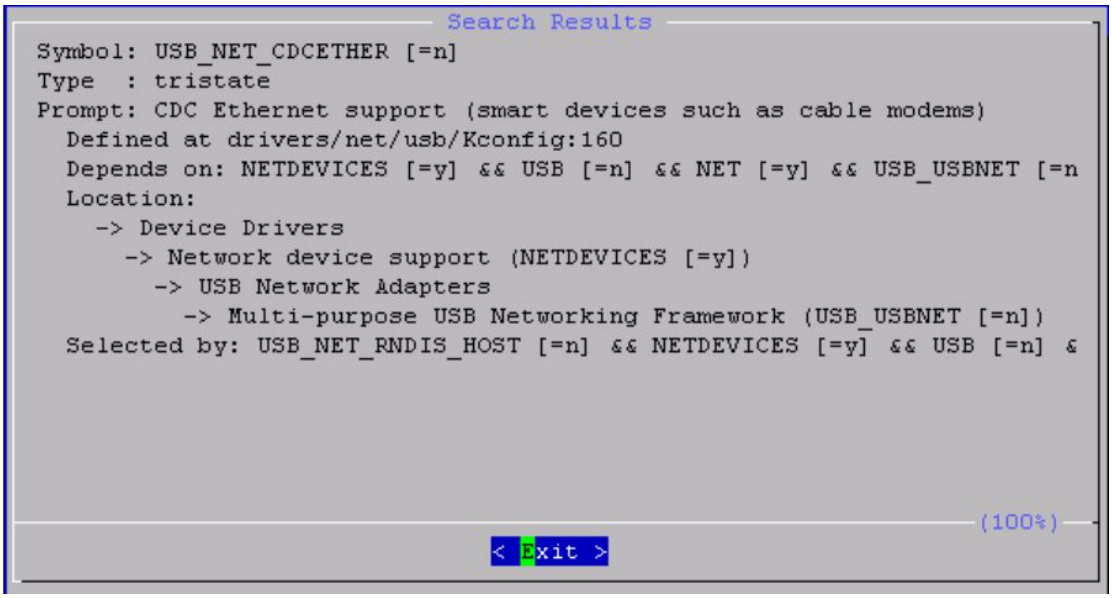
```
*/
return 0;
error:
if (info->data) {
usb_set_intfdata(info->data, NULL);
usb_driver_release_interface(driver_of(intf), info->data);
}
return status;
}
int usbnet_cdc_bind(struct usbnet *dev, struct usb_interface *intf)
{
int status;
struct cdc_state    *info = (void *) &dev->data;
BUILD_BUG_ON((sizeof(((struct usbnet *)0)->data)< sizeof(struct cdc_state)));
status = usbnet_generic_cdc_bind(dev, intf);
if(status < 0)
      {
            printk("usbnet_generic_cdc_bind failure\n");
            status = usbnet_get_endpoints(dev,intf);

            if(status < 0)
            {
                  printk("usbnet_get_endpoints failure\n");
                  return status;
            }
      }
      if(info->ether)
      {
            status = usbnet_get_ethernet_addr(dev,info->ether->iMACAddress);
            if(status <0)
            {
                  printk("usbnet_get_ethernet_addr failure\n");
                  usb_set_intfdata(info->data,NULL);
                  usb_driver_release_interface(driver_of(intf),info->data);
                  return status;
            }
      }

/* FIXME cdc-ether has some multicast code too, though it complains
* in routine cases.      info->ether describes the multicast support.
* Implement that here, manipulating the cdc filter as needed.
*/
return 0;
}
```

## 2、 提示

**2.1** Linux 3.8 版本的内核已经支持模块的 USB 驱动。 USB 转串口的驱动的内核配置同上文描述，网络数据接口需要编译文件 drivers/net/usb/qmi_wwan.c。

**2.2** 驱动加载后， USB 转串口驱动的设备名字一般为 ttyUSBx(x 为某个数字，通常为 ttyUSB0、ttyUSB1、ttyUSB2)。网络设备的名字根据内核版本显示不同的名字，需要使用 ifconfig 命令对比加载前后网络设备的变化来确定具体的设备名字。

**2.3** 观察对应目录下的 的 Makefile 文件可以看到编译 文件依赖的配置变量，比如 drivers/net/usb/cdc_ether.c 文件依赖 CONFIG_USB_NET_CDCETHER，在 menuconfig 界面，可以搜索 USB_NET_CDCETHER，观察配置该变量的依赖关系，从而明确如何才能配置这个变量。
以我们的系统为例，搜索到的结果如下图：



图中提示，该变量的配置路径为：

```
Location:
  -> Device Drivers
    -> Network device support (NETDEVICES [=y])
      -> USB Network Adapters
        -> Multi-purpose USB Networking Framework (USB_USBNET [=n])
```

依赖关系为：

```
  Depends on: NETDEVICES [=y] && USB [=n] && NET [=y] && USB_USBNET [=n
```

因此，需要将 NETDEVICES、USB、NET、USB_USBNET 四个变量都配置成 y，才能在 menuconfig 中找到配置 USB_NET_CDCETHER 的项。

**2.4** B78 系列模块使用的版本升级端口为 USB 转串口，驱动加载方法同上文描述，只是 USB 的 PID 有变化，下载端口的 VID 为 0x19D2，PID 为 0x0256。