VxWorks 配置多网口

概述

本文提供在系统运行中对网口进行配置的方法。

注意

开发环境：VxWorks6.9.4，Workbench3.3.5。

1. 之前小编网上找相关资料时，有博主说使用新增网口不能与已存在的网口处于同一网段。不过经过小编的测试，两个网口 ip 可以处在同一网段。

```
-> ipAttach 3,"gei"
value = 0 = 0x0
-> ifconfig "gei3 192.168.10.200 netmask 255.255.255.0 up"
value = 0 = 0x0
-> ifconfig
lo0     Link type:Local loopback  Queue:none
        inet 127.0.0.1  mask 255.255.255.255
        UP RUNNING LOOPBACK MULTICAST NOARP ALLMULTI
        MTU:1500  metric:1  VR:0  ifindex:1
        RX packets:63 mcast:0 errors:0 dropped:0
        TX packets:63 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:2778  TX bytes:2778

gei2    Link type:Ethernet  HWaddr 00:30:64:6a:88:10  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 192.168.10.144  mask 255.255.255.0  broadcast 192.168.10.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:2
        RX packets:51 mcast:0 errors:0 dropped:0
        TX packets:18 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:3181  TX bytes:1098

gei3    Link type:Ethernet  HWaddr 00:00:64:6a:88:13  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 192.168.10.200  mask 255.255.255.0  broadcast 192.168.10.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:3
        RX packets:11 mcast:0 errors:0 dropped:6
        TX packets:1 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:692  TX bytes:42
```

使用命令行测试网络连接正常。

```
C:\Users\rent>ping 192.168.10.200

正在 Ping 192.168.10.200 具有 32 字节的数据:
来自 192.168.10.200 的回复: 字节=32 时间=1ms TTL=64
来自 192.168.10.200 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.200 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.200 的回复: 字节=32 时间<1ms TTL=64

192.168.10.200 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 1ms, 平均 = 0ms

C:\Users\rent>ping 192.168.10.144

正在 Ping 192.168.10.144 具有 32 字节的数据:
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64

192.168.10.144 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\rent>
        半:
```

2. 若网口没有连接，状态显示会缺少 RUNNING 字符。

```
-> ifconfig
lo0     Link type:Local loopback  Queue:none
        inet 127.0.0.1  mask 255.255.255.255
        UP RUNNING LOOPBACK MULTICAST NOARP ALLMULTI
        MTU:1500  metric:1  VR:0  ifindex:1
        RX packets:87 mcast:0 errors:0 dropped:0
        TX packets:88 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:4850  TX bytes:4898

gei2    Link type:Ethernet  HWaddr 00:30:64:6a:88:10  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 192.168.10.144  mask 255.255.255.0  broadcast 192.168.10.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:2
        RX packets:709 mcast:0 errors:0 dropped:0
        TX packets:35 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:43k  TX bytes:3267

gei3    Link type:Ethernet  HWaddr 00:00:64:6a:88:13  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 192.168.10.200  mask 255.255.255.0  broadcast 192.168.10.255
        UP SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:3
        RX packets:636 mcast:0 errors:0 dropped:6
        TX packets:6 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:38k  TX bytes:398

value = 0 = 0x0
->
```

验证

启动目标机，输入命令 ifconfig，查看当前系统只有一个网口（gei2）。

```
-> ifconfig
lo0     Link type:Local loopback  Queue:none
        inet 127.0.0.1  mask 255.255.255.255
        UP RUNNING LOOPBACK MULTICAST NOARP ALLMULTI
        MTU:1500  metric:1  VR:0  ifindex:1
        RX packets:427 mcast:0 errors:0 dropped:0
        TX packets:427 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:37k  TX bytes:37k

gei2    Link type:Ethernet  HWaddr 00:30:64:6a:88:10  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 192.168.10.144  mask 255.255.255.0  broadcast 192.168.10.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:2
        RX packets:1130 mcast:0 errors:0 dropped:0
        TX packets:133 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:72k  TX bytes:27k

value = 0 = 0x0
->
```

打开 datasheet，根据以太网的信息描述，实际目标机的网卡并不止一个。

- **连接**

PMC/XMC

可选PCI 64位/66MHz PMC槽或PCI-Express x1 XMC槽

以太网

一个Intel® I217LM控制器，支持一个千兆以太网前面板端口和Intel® AMT 9.0
三个Intel® I210控制器，支持一个千兆以太网前面板端口和两个后面板端口
额外两个Intel® 82574L控制器，支持两个千兆网

打开 bsp，可以看到网络驱动为 GEI825XX_VXB_END，挂在 VxBus 下。

```
config.h
# endif /* INCLUDE_SIO_UTILS */

/* Network driver options: VxBus drivers */

# ifdef INCLUDE_END
#undef  INCLUDE_AM79C97X_VXB_END
#undef  INCLUDE_AN983_VXB_END
#undef  INCLUDE_FEI8255X_VXB_END
#define INCLUDE_GEI825XX_VXB_END
#undef  INCLUDE_MVYUKONII_VXB_END
#undef  INCLUDE_MVYUKON_VXB_END
#undef  INCLUDE_NS8381X_VXB_END
#undef  INCLUDE_RTL8139_VXB_END
#undef  INCLUDE_RTL8169_VXB_END
#undef  INCLUDE_TC3C905_VXB_END
#undef  INCLUDE_NE2000_VXB_END

/* PHY and MII bus support */

#undef  INCLUDE_DM9191PHY
#undef  INCLUDE_LXT972PHY
#undef  INCLUDE_MV88E1X11PHY
#undef  INCLUDE_RTL8201PHY
#undef  INCLUDE_RTL8169PHY
#undef  INCLUDE_VSC82XXPHY
# endif /* INCLUDE_END */
```

打开镜像工程，添加组件 INCLUDE_VXBUS_SHOW。用于查看 vxBus 相关信息。

Components

Component Configuration

Description	Name	Type	Value
Inter-Integrated Circuit Bus	INCLUDE_I2C_BUS		
PCI Bus Auto Configuration Routine	**INCLUDE_PCI_BUS_AUTOCONF**		
PCI Bus Show Routines	**INCLUDE_PCI_BUS_SHOW**		
PCI Bus legacy Auto Configuration	**INCLUDE_PCI_OLD_CONFIG_ROUTINES**		
Peripheral Component Interconnect	**INCLUDE_PCI_BUS**		
Processor Local Bus (default)	**INCLUDE_PLB_BUS**		
SD Bus	INCLUDE_SD_BUS		
Serial Peripheral Interface Bus	INCLUDE_SPI_BUS		
vxBus subsystem (default)	**INCLUDE_VXBUS**		
vxBus subsystem show routines	**INCLUDE_VXBUS_SHOW**		
Device Drivers	**FOLDER_DRIVERS**		
Hardware Interface Modules	**FOLDER_HWIF**		
buses	**FOLDER_BUSES**		
itl_haswell_64 BSP configuration optio	**FOLDER_BSP_CONFIG**		
memory (default)	**FOLDER_MEMORY**		
peripherals	**FOLDER_PERIPHERALS**		
General mkboot module for Intel Archite	INCLUDE_MKBOOT		
obsolete components	FOLDER_OBSOLETE		
operating system components (default)	**FOLDER_OS**		

输入命令 vxBusShow，查看当前 vxBus 相关信息。此时网络驱动已经注册到设备上了。

```
-> vxBusShow
Registered Bus Types:
  USB-EHCI_Bus @ 0xffff80000013fc0
  USB-Host_Bus @ 0xffffffff806cf640
  USB-HUB_Bus @ 0xffffffff806cf600
  MII_Bus @ 0xffffffff806caa60
  PCI_Bus @ 0xffffffff806ca620
  PLB_Bus @ 0xffffffff806ca6e0

Registered Device Drivers:
  vxbUsbKeyboard at 0xffff800000049f00 on bus USB-HUB_Bus, funcs @ 0xffffffff806cf590
  vxbUsbBulkClass at 0xffff800000014100 on bus USB-HUB_Bus, funcs @ 0xffffffff806cf590
  vxbUsbHubClass at 0xffffffff82064b20 on bus USB-HUB_Bus, funcs @ 0xffffffff806cf590
  pentiumPci at 0xffffffff806c47e0 on bus PLB_Bus, funcs @ 0xffffffff806c4900
  mpApic at 0xffffffff806c98c0 on bus PLB_Bus, funcs @ 0xffffffff806c9a00
  mc146818Rtc at 0xffffffff806ca0c0 on bus PLB_Bus, funcs @ 0xffffffff806ca120
  loApicTimer at 0xffffffff806ca360 on bus PLB_Bus, funcs @ 0xffffffff806ca3c0
  loApicIntr at 0xffffffff806c9bc0 on bus PLB_Bus, funcs @ 0xffffffff806c9da0
  ioApicIntr at 0xffffffff806c9a40 on bus PLB_Bus, funcs @ 0xffffffff806c9b80
  iaTimestamp at 0xffffffff806ca2a0 on bus PLB_Bus, funcs @ 0xffffffff806ca300
  ichAta at 0xffffffff806cabe0 on bus PCI_Bus, funcs @ 0xffffffff806cac80
  intelAhciSata at 0xffffffff806caee0 on bus PLB_Bus, funcs @ 0xffffffff806cb050
  intelAhciSata at 0xffffffff806cafe0 on bus PCI_Bus, funcs @ 0xffffffff806cb050
  iaHpetTimerDev at 0xffffffff806ca420 on bus PLB_Bus, funcs @ 0xffffffff806ca480
  i8253TimerDev at 0xffffffff806ca180 on bus PLB_Bus, funcs @ 0xffffffff806ca240
  i8253TimerDev at 0xffffffff806ca1e0 on bus MF_Bus, funcs @ 0xffffffff806ca240
  ahciSata at 0xffffffff806cad40 on bus PLB_Bus, funcs @ 0xffffffff806cae90
  ahciSata at 0xffffffff806cae20 on bus PCI_Bus, funcs @ 0xffffffff806cae90
  vxbUsbEhciHub at 0xffffffff806cf280 on bus USB-EHCI_Bus, funcs @ 0xffffffff806cf510
  vxbPlbUsbEhci at 0xffffffff806cf220 on bus PLB_Bus, funcs @ 0xffffffff806cf3f0
  vxbPciUsbEhci at 0xffffffff806cf1a0 on bus PCI_Bus, funcs @ 0xffffffff806cf2e0
  ns16550 at 0xffffffff806c9e60 on bus PLB_Bus, funcs @ 0xffffffff806c9fb0
  ns16550 at 0xffffffff806c9f40 on bus PCI_Bus, funcs @ 0xffffffff806c9fb0
  i8042Kbd at 0xffffffff806c93a0 on bus PLB_Bus, funcs @ 0xffffffff806c9480
  genericPhy at 0xffffffff806cab00 on bus MII_Bus, funcs @ 0xffffffff806cab60
  miiBus at 0xffffffff806ca9e0 on bus PCI_Bus, funcs @ 0xffffffff806caaa0
  miiBus at 0xffffffff806ca980 on bus PLB_Bus, funcs @ 0xffffffff806caaa0
  gei at 0xffffffff806c4300 on bus PCI_Bus, funcs @ 0xffffffff806c43f0
  m6845Vga at 0xffffffff806c94a0 on bus PLB_Bus, funcs @ 0xffffffff806c9560
  plbCtlr at 0xffffffff806ca680 on bus PLB_Bus, funcs @ 0xffffffff806ca8c0
```

向下翻页，看到系统包含四个网口设备。

```
PCI_Bus @ 0xffff80000000c960 with bridge @ 0xffff80000000c1a0
  Device Instances:
      gei unit 0 on PCI_Bus @ 0xffff80000000e0a0 with busInfo 0x0000000000000000
      gei unit 1 on PCI_Bus @ 0xffff80000000e500 with busInfo 0x0000000000000000
      gei unit 2 on PCI_Bus @ 0xffff80000000f220 with busInfo 0x0000000000000000
      vxbPciUsbEhci unit 1 on PCI_Bus @ 0xffff80000000f450 with busInfo 0xffff800000267160
      gei unit 3 on PCI_Bus @ 0xffff80000000ff40 with busInfo 0x0000000000000000
      vxbPciUsbEhci unit 0 on PCI_Bus @ 0xffff800000010a30 with busInfo 0xffff80000026ff70
      ahciSata unit 0 on PCI_Bus @ 0xffff800000010e90 with busInfo 0x0000000000000000
      miiBus unit 0 on PCI_Bus @ 0xffff8000002db7e0 with busInfo 0xffff8000002dc5e0
```

此时，我们需要配置其余的网口设备。

打开帮助文档，5.3.4 章节提供了如何在运行时配置网络接口的方法。

5.3.4 At Run Time

If you are not ready to configure the interface at build time, you can configure it at run time. This procedure consists of two steps:

1. Attaching a protocol.
2. Configuring the address and subnet mask.

Using a Shell Command

To perform these steps, run an **ipAttach** shell command on the target, followed by an **ifconfig**. For example:

```
[vxWorks *] # ipAttach 1,fei
[vxWorks *] # ifconfig fei1 10.0.0.2 netmask 255.255.255.0 up
```

The parameters for the **ifconfig** command are specified in ifconfig.

Using an API

Two libraries contain APIs for run-time configuration of network interfaces--**ifconfig** and **ifLib**.

For example, you can assign a network address to an interface, using code such

```
ifconfig( inet add "fei1 10.0.0.2 netmask 255.255.255.0 " )
```

按照提供的方法，执行命令：

ipAttach 3,"gei"和 ifconfig "gei3 10.0.0.2 netmask 255.255.255.0 up"，配置 gei3 网口。
执行完成后，执行 ifconfig，可以看到 gei3 已经成功配置。

```
-> ipAttach 3,"gei"
value = 0 = 0x0
-> ifconfig "gei3 10.0.0.2 netmask 255.255.255.0 up"
value = 0 = 0x0
-> ifconfig
lo0     Link type:Local loopback  Queue:none
        inet 127.0.0.1  mask 255.255.255.255
        UP RUNNING LOOPBACK MULTICAST NOARP ALLMULTI
        MTU:1500  metric:1  VR:0  ifindex:1
        RX packets:63 mcast:0 errors:0 dropped:0
        TX packets:63 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:2772  TX bytes:2772

gei2    Link type:Ethernet  HWaddr 00:30:64:6a:88:10  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 192.168.10.144  mask 255.255.255.0  broadcast 192.168.10.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:2
        RX packets:116 mcast:0 errors:0 dropped:0
        TX packets:20 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:7310  TX bytes:1225

gei3    Link type:Ethernet  HWaddr 00:00:64:6a:88:13  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 10.0.0.2  mask 255.255.255.0  broadcast 10.0.0.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:3
        RX packets:24 mcast:0 errors:0 dropped:17
        TX packets:1 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:1627  TX bytes:42
```

使用任务管理器测试两个网口均能正常联通。

```
C:\Users\rent>ping 10.0.0.2

正在 Ping 10.0.0.2 具有 32 字节的数据:
来自 10.0.0.2 的回复: 字节=32 时间=1ms TTL=64
来自 10.0.0.2 的回复: 字节=32 时间<1ms TTL=64
来自 10.0.0.2 的回复: 字节=32 时间<1ms TTL=64
来自 10.0.0.2 的回复: 字节=32 时间<1ms TTL=64

10.0.0.2 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 1ms, 平均 = 0ms

C:\Users\rent>ping 192.168.10.144

正在 Ping 192.168.10.144 具有 32 字节的数据:
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.144 的回复: 字节=32 时间<1ms TTL=64

192.168.10.144 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\rent>_
```

同样，我们可以为 gei3 网口添加 ip “192.168.100.7”。

```
-> ifAddrAdd "gei3", "192.168.100.7","192.168.100.0",0xffffff00
value = 0 = 0x0
-> ifconfig
lo0     Link type:Local loopback  Queue:none
        inet 127.0.0.1  mask 255.255.255.255
        UP RUNNING LOOPBACK MULTICAST NOARP ALLMULTI
        MTU:1500  metric:1  VR:0  ifindex:1
        RX packets:119 mcast:0 errors:0 dropped:0
        TX packets:119 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:6374  TX bytes:6374

gei2    Link type:Ethernet  HWaddr 00:30:64:6a:88:10  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 192.168.10.144  mask 255.255.255.0  broadcast 192.168.10.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:2
        RX packets:1145 mcast:0 errors:0 dropped:0
        TX packets:52 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:69k  TX bytes:4410

gei3    Link type:Ethernet  HWaddr 00:00:64:6a:88:13  Queue:none
        capabilities: TXCSUM TX6CSUM
        inet 10.0.0.2  mask 255.255.255.0  broadcast 10.0.0.255
        inet 192.168.100.7  mask 255.255.255.0  broadcast 192.168.100.255
        UP RUNNING SIMPLEX BROADCAST MULTICAST
        MTU:1500  metric:1  VR:0  ifindex:3
        RX packets:1025 mcast:0 errors:0 dropped:17
        TX packets:8 mcast:0 errors:0
        collisions:0 unsupported proto:0
        RX bytes:62k  TX bytes:482

value = 0 = 0x0
->
```

使用命令行测试网络连接正常。

```
C:\Users\rent>ping 192.168.100.7

正在 Ping 192.168.100.7 具有 32 字节的数据:
来自 192.168.100.7 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.7 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.7 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.7 的回复: 字节=32 时间<1ms TTL=64

192.168.100.7 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\rent>_
        半:
```