COMPREHENSIVE SERVICES

We offer competitive repair and calibration services, as well as easily accessible documentation and free downloadable resources.

SELL YOUR SURPLUS

We buy new, used, decommissioned, and surplus parts from every NI series. We work out the best solution to suit your individual needs.

Sell For Cash Get Credit Receive a Trade-In Deal

OBSOLETE NI HARDWARE IN STOCK & READY TO SHIP

We stock **New, New Surplus, Refurbished,** and **Reconditioned** NI Hardware.

APEX WAVES

Bridging the gap *between the manufacturer and your legacy test system.*

1-800-915-6216

www.apexwaves.com

sales@apexwaves.com

Request a Quote CLICK HERE ***PCI-485-2***

USING PCI SERIAL WITH LINUX

This document contains instructions to help you install and configure the National Instruments serial hardware for Linux. This document includes information about the PCI-232/2, PCI-232/4, PCI-232/8, PCI-485/2, PCI-485/4, PCI-485/8, PCI-232/2 Isolated, PCI-232/4 Isolated, PCI-485/2 Isolated, and PCI-485/4 Isolated interfaces.

This document assumes that you are already familiar with Linux.

Contents

Related Documentation

The following documents contain information that you might find helpful as you read this document:

- *Linux Serial-Programming-HOWTO* by Peter Baumann. You can find the latest version of this document at the following locations:

  `ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/Serial-Programming-HOWTO`

  `http://metalab.unc.edu/LDP/HOWTO/Serial-Programming-HOWTO.html`

- *Linux Serial-HOWTO* by David Lawyer. You can find the latest version of this document at the following locations:

  `ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/Serial-HOWTO`

  `http://metalab.unc.edu/LDP/HOWTO/Serial-HOWTO.html`

Contributions

Thanks to Vern Howie for providing suggestions and examples from his serial suite. Also, thanks to David Lawyer, Greg Hankins, and Peter Baumann for providing so much information in their HOWTOs.

Gather What You Need to Get Started

Before you install your PCI serial interface for Linux, make sure you have the following:

- Linux kernel version 2.2.3 or later. The product has been thoroughly tested with kernel version 2.2.3; however, the product might work with earlier kernel versions.

  If you do not have kernel version 2.2.3 or later, or if you do not have the following options already compiled into your kernel, you need to recompile you kernel. Include the following options when you configure and recompile the kernel using `make menuconfig`.

a. General Setup
   - PCI Support
   - PCI Quirks
   - Backward-compatible `/proc/pci`

b. Character Devices
   - Standard/generic dumb serial support
   - Extended dumb serial driver options
   - Support more than four serial ports
   - Support for sharing serial interrupts

- `setserial` 2.14 or later. To find the version of `setserial`, enter the following:

  ```
  linux# setserial -V
  ```

- PCI Utilities 1.10 or later. You can download PCI Utilities 1.10 from `ftp://metalab.unc.edu/pub/Linux/hardware`
- `PCI-SERIAL.tar.gz`. You can download this file from the National Instruments FTP site at `ftp://ftp.natinst.com/support/ind_comm/serial/linux/`

  After you have the file, extract and unzip it by entering the following:

  ```
  linux#  tar zxvf PCI-SERIAL.tar.gz
  ```

  The `tar` command extracts and unzips `PCI-SERIAL.tar.gz` and creates the sub-directory `PCI-SERIAL`. Enter the following to make sure all necessary files are included:

  ```
  linux#                cd PCI-SERIAL
  linux PCI-SERIAL#     ls
    FIFOtrigger  intenable  serialtest
    FIFOtrigger_pci.c  interrupt_enable.c serialtest.c
    Clock_speed.c  rs485  termios_program.c
    clockspeed  rs485.c
  ```

- Configure your BIOS to include a Plug and Play aware OS.
- You need superuser privileges to do most of the steps and program segments in this document.

Quick Start

This section is for experienced Linux users who are familiar with the `lspci` and `setserial` tools. If you are not familiar with either of these tools or if you require a detailed explanation of the steps, skip to the next section, *Setup*.

1. If you do not have enough available serial devices (`/dev/ttyS*`) for each port on your multiport interface, create a new serial device by entering the following:

   ```
   linux# cd /dev
   linux /dev# ./MAKEDEV ttyS<port number>
   ```

2. Find the port address, IRQ, and memory assignment of your PCI serial interface by enter the following:

   ```
   linux# lspci -v -n -d 1093:*
   ```

3. Assign the serial driver to your devices. Make sure you precede the port addresses with `0x`.

   - For a PCI-232 interface, enter the following:

     ```
     linux# setserial /dev/ttyS<port number> uart
            16550a port <port address> irq <irq>
            ^fourport
     ```

   - For a PCI-485 interface, which supports a higher `baud_base` of 460.8K at startup, enter the following:

     ```
     linux# setserial /dev/ttyS<port number> uart
            16550a port <port address> irq <irq>
            baud_base 460800 ^fourport
     ```

4. Enable the PCI interrupt on your interface. To use `intenable` (from the `PCI-SERIAL` directory) to enable interrupts on your PCI serial interface, enter the following:

   ```
   linux PCI-SERIAL# ./intenable <pci memory address
                     found in lspci>
   ```

5. If you have a PCI-485 interface, set the transceiver mode for each serial port. Refer to the section *Select Transceiver Mode* for more information about selecting a PCI-485 transceiver mode.

6. After you connect a cable between the two ports, test the setup by running `serialtest` (from the `PCI-SERIAL` directory).

   ```
   linux#   ./serialtest <receive port number> <transmit
            port number>
   ```

Setup

After you install the serial hardware (as shown in your PCI serial getting started manual), follow these steps to set up the interface.

Create Devices

Create a device for each port on your multiport interface. You only need to do this step once. *Port address* is the I/O address of the device, and *port number* is the device/serial port number of the port. Port number is used in the following context: `ttyS<port number>`. Since serial ports built into your computer are typically named from `/dev/ttyS0` to `/dev/ttyS3`, the port number you choose needs to be four or greater.

Enter the `/dev` directory, then use the `MAKEDEV` script to create a device for each serial port on the interface by entering the following.

```
linux# cd /dev
linux /dev# ./MAKEDEV ttyS<port number>
```

MAKEDEV Example

Enter the following to make the devices for a two-port interface:

```
linux# cd /dev
linux /dev# ./MAKEDEV ttyS4
linux /dev# ./MAKEDEV ttyS5
```

Find Interface Information

Follow these instructions to find the port address, IRQ, and memory assignment of your PCI serial interface.

Note Repeat this section each time you add another interface or physical device to the computer.

Note This step may be done in normal user mode.

1. Use `lspci (`a command that displays information about the PCI bus) with the `-v` option (be verbose), the `-n` option (show PCI vendor and device codes as numbers), and the `-d 1093:*` option (display information only on devices with a National Instruments Vendor ID of 1093), to find the resource information of your serial interface.

   ```
   linux# lspci -v -n -d 1093:*
   ```

   Something similar to the following should appear. In this output, the IRQ is 11; the memory location is 0xdff80, and the port addresses are 0xdff0 and 0xdfe0.

```
00:0a.0 Class ff00: 1093:d140 (rev 01)
    Flags: medium devsel, IRQ 11
    Memory at 000dff80 (low-1M, non-prefetchable)
    I/O ports at dff0
    I/O ports at dfe0
```

Write down the IRQ, memory location, and all the I/O port addresses for your computer.

Note PCI Eight-Port Users—The PCI eight-port interfaces show only one I/O port address listing. The addresses of the other seven I/O ports are calculated by adding eight to the previous port address, ($n \times 8$ + I/O port) for $0 < n < 8$. The `lspci` call displays something similar to the following:

```
00:0a.0 Class ff00: 1093:d150 (rev 01)
    Flags: medium devsel, IRQ 11
    Memory at 000dff80 (low-1M, non-prefetchable)
    I/O ports at df00
```

Assign Serial Driver

Assign the serial driver to your devices.

Note You need to repeat this step each time you restart your computer or until you set up your `/etc/rc.d/rc.serial` file. (Refer to the section *Sample /etc/rc.d/rc.serial File* for more information on setting up the `/etc/rc.d/rc.serial` file.)

Enter the following to use `setserial` to tell the kernel each device's UART, port address, and IRQ. Use information returned from the `lspci` output, and remember to precede the port address with `0x`.

```
linux#  setserial /dev/ttyS<port number> uart 16550A
        port <port address> irq <irq> ^fourport
```

Note The `^fourport` flag is required regardless of how many ports you have on your interface. The `^fourport` flag tells the serial driver that you are not using an AST four-port interface.

Caution Using an invalid port can lock up your machine.

setserial Example

Enter the following to assign the serial driver to your devices for the values in the above two-port `lscpi` output:

```
linux#  setserial /dev/ttyS4 uart 16550A port 0xdff0 irq
        11 ^fourport
```

```
linux#  setserial /dev/ttyS5 uart 16550A port 0xdfe0 irq
        11 ^fourport
```

Assign Serial Drivers Using PCI-485

Since the PCI-485 supports a higher `baud_base` of 460.8K at start-up, also enter the following at the linux prompt:

```
linux#  setserial /dev/ttyS<port number> baud_base
460800
```

For example, enter the following to assign the serial driver to `/dev/ttyS4` for a PCI-485 serial port at port address 0xdfe0 and IRQ 11.

```
linux#  setserial /dev/ttyS4 uart 16550A port 0xdfe0 irq
        11 ^fourport
linux#  setserial /dev/ttyS4 baud_base 460800
```

Assign Serial Drivers Using PCI Eight-Port Interfaces

Enter the following to assign the serial driver to your devices for the PCI eight-port `lspci` output. Notice that the entries for port are consecutive and are separated by exactly 8 bytes.

```
linux# setserial /dev/ttyS4 uart 16550A port 0xdf00 irq
11 ^fourport
linux# setserial /dev/ttyS5 uart 16550A port 0xdf08 irq
11 ^fourport
linux# setserial /dev/ttyS6 uart 16550A port 0xdf10 irq
11 ^fourport
linux# setserial /dev/ttyS7 uart 16550A port 0xdf18 irq
11 ^fourport
linux# setserial /dev/ttyS8 uart 16550A port 0xdf20 irq
11 ^fourport
linux# setserial /dev/ttyS9 uart 16550A port 0xdf28 irq
11 ^fourport
linux# setserial /dev/ttyS10 uart 16550A port 0xdf30 irq
11 ^fourport
linux# setserial /dev/ttyS11 uart 16550A port 0xdf38 irq
11 ^fourport
```

Enable PCI Interrupt

To enable the PCI interrupt on your serial interface, use `intenable`, provided in the `PCI-SERIAL` directory. Run `intenable` each time you restart your computer. If you have more than one PCI serial interface, rerun `intenable` with another PCI memory address specified in the command line. Enter the following to use `intenable`:

```
linux PCI-SERIAL# ./intenable <PCI memory address found
                  in lspci>
```

For example, enter the following to enable interrupts for a PCI serial interface with memory at 0x000dff80:

```
linux PCI-SERIAL# ./intenable dff80
```

Configuration

View Your Hardware Resources

To see what system resources your serial interface is using, use the `setserial` command, as follows:

```
linux# setserial -gv /dev/ttyS<port number>
```

For the `lspci` and `setserial` examples, something similar to the following should appear:

```
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
/dev/ttyS1, UART: unknown, Port: 0x02f8, IRQ: 3
/dev/ttyS2, UART: unknown, Port: 0x03e8, IRQ: 4
/dev/ttyS3, UART: unknown, Port: 0x02e8, IRQ: 3
/dev/ttyS4, UART: 16550A, Port: 0xdff0, IRQ: 11
/dev/ttyS5, UART: 16550A, Port: 0xdfe0, IRQ: 11
```

Enable FIFO Buffers

Use `FIFOtrigger` (from the `PCI-SERIAL` directory) to enable the receive and transmit FIFOs and to set the trigger levels of these FIFOs. `FIFOtrigger` enables the FIFO of only one serial port. To enable the FIFO for your other serial ports, rerun `FIFOtrigger` with a different serial port number specified in the command line. Enter the following to use `FIFOtrigger`:

```
linux PCI-SERIAL#./FIFOtrigger <port number>
                  <rx_trigger> <tx_trigger>
```

Table 1. tx_trigger Values

Transmit FIFO Trigger Level	tx_trigger
8	0x00
16	0x10
32	0x20
56	0x30

Table 2. rx_trigger Values

Receive FIFO Trigger Level	rx_trigger
8	0x00
16	0x40
56	0x80
60	0xC0

The hardware issues a *transmit empty* interrupt when the number of characters in the transmit FIFO falls below the trigger level. Also, the hardware issues a *receive full* interrupt when the number of characters in the receive FIFO rises above the trigger level. For more information on FIFO buffers, refer to your PCI serial getting started manual.

FIFO Example

Enter the following to set the receive FIFO trigger level to 56 and the transmit trigger level to 32 for `/dev/ttyS5`:

```
linux PCI-SERIAL# ./FIFOtrigger 5 0x80 0x20
```

Configure struct termios

Every serial port has an associated `struct termios`. By using this `struct termios` in a program, you can set the baud rate, character size (number of data bits), parity, control characters, flow control, and input and output mode. For more information about the termios structure, refer to the `termios man page`. To view the `termios man page`, enter the following:

```
linux# man termios
```

To configure your serial port, use a program segment similar to the `termios_program.c` in your `PCI-SERIAL` directory.

Configure PCI-485

If you are using a PCI-485 interface, you can select the transceiver mode for each device and select the lower baud rates listed in `termios_program.c` provided in your `PCI-SERIAL` directory.

Select Transceiver Mode

Use the `rs485` program (from the `PCI-SERIAL` directory) to select the transceiver mode. `rs485` sets the transceiver mode for only one serial port. To set the transceiver mode for other serial ports, rerun `rs485` with a different serial port number specified in the command line. For more information on the transceiver control modes, refer to your serial getting started manual. If you do not know which transceiver mode to use, choose Four-Wire Mode.

Table 3. Transceiver Mode Values

Transceiver Mode	Mode
Four-Wire Mode	0
Two-Wire Mode: DTR with echo	1
Two-Wire Mode: DTR controlled	2
Two-Wire Mode: TXRDY auto control	3

Enter the following to use `rs485`:

```
linux PCI-SERIAL# ./rs485 <port number> <mode>
```

rs485 Example

Enter the following to select Four-Wire Mode for `/dev/ttyS5`:

```
linux PCI-SERIAL# ./rs485 5 0
```

Lower Baud Rate Selection

To use a baud rate lower than 200, use `setserial` and the `clockspeed` program (from the `PCI-SERIAL` directory) to lower the baud base of a single serial port from 460800 to 115200. After changing the baud base to 115200, the max baud rate you can achieve is 115200 baud (until you change the baud base back to 460800).

After you change the `baud_base` of the serial port using `setserial` and run the `clockspeed` program, you can use the lower baud rate flags in your programs. The `clockspeed` program changes the baud base of only

one serial port. To change the baud base of your other serial ports, rerun `clockspeed` with a different serial port number specified in the command line. Enter the following to set the baud base. (In this example, *high* equals a baud base of 460800 and *low* equals a baud base of 115200):

```
linux# setserial /dev/ttyS<port number> baud_base 115200
linux PCI-SERIAL#./clockspeed <port number> <"high" or
                 "low">
```

For example, enter the following to select a baud base of 115200 for `/dev/ttyS4` and for `/dev/ttyS5`:

```
linux# setserial /dev/ttyS4 baud_base 115200
linux# setserial /dev/ttyS5 baud_base 115200
linux PCI-SERIAL# ./clockspeed 4 low
linux PCI-SERIAL# ./clockspeed 5 low
```

Enter the following to change the baud base back to 460800 for `/dev/ttyS4` and for `/dev/ttyS5`:

```
linux# setserial /dev/ttyS4 baud_base 460800
linux# setserial /dev/ttyS5 baud_base 460800
linux PCI-SERIAL# ./clockspeed 4 high
linux PCI-SERIAL# ./clockspeed 5 high
```

Test the Setup

After you connect the cables to the port (as shown in your PCI serial getting started manual), run the `serialtest` program (from the `PCI-SERIAL` directory) to verify your setup. Make sure you specify two different ports for the `serialtest` program, as shown in the following:

```
linux PCI-SERIAL#  ./serialtest <receive port number>
                   <transmit port number>
```

If the test is successful, it displays a `SUCCESS` message. If the test hangs, press <ctrl-c> to exit the program, and continue to the next section, *Troubleshooting and Common Questions*.

To test `/dev/ttyS4` and `/dev/ttyS5`, connect a cable between the two ports and enter the following:

```
linux PCI-SERIAL# ./serialtest 4 5
```

Troubleshooting and Common Questions

Error Codes

This section lists possible error codes and solutions.

Error Code	**/dev/ttyS<*port number*>: no such file or directory**
Solution	The /dev/ttyS<*port number*> does not exist in the /dev directory. Enter the following to create the device: **linux#** cd /dev **linux /dev#** ./MAKEDEV ttyS<*port number*>

Error Code	**Couldn't change i/o privilege level: Operation not permitted**
Solution	The program requires superuser privileges. Either exit and log in as root, or enter the following: **linux$** su **Password:** <*enter the root password*> **linux#** <*run the program*>

Error Code	setserial: **Cannot set serial info: Address already in use**
Solution	Make sure you are entering the correct port address into setserial. Also, make sure you are entering 0x if you are specifying a hex number. **linux#** setserial /dev/ttyS<*port number*> uart 16550a port 0x<*port address*> irq <*irq*> ^fourport

Error Code	intenable: **Can't open /dev/mem: Permission denied**
Solution	The program requires superuser privileges. Either exit and log in as root, or enter the following: **linux$** su **Password:** <*enter the root password*> **linux#** ./intenable <*PCI memory*>

Error Code	intenable: **ERROR: Initial value of interrupt enable register not equal to 0xC0C**
Solution	Make sure that you are providing the program with the correct memory address. Enter the following to check for the memory address: **linux#** lspci –v –n –d 1093:* 00:0a.0 Class ff00: 1093:d150 (rev 01) Flags: medium devsel, IRQ 11 Enter this address! **-->Memory at 000dff80** (low-1M, non-prefetchable) I/O ports at dff0 I/O ports at dfe0 I/O ports at dfa8 I/O ports at dfa0 **linux#** ./intenable dff80

Error Code	rs485: **ERROR: Couldn't write to /dev/ttyS<*port number*>'s scratch register**
Solution	Make sure the device was configured correctly in setserial. Enter the following: **linux#** setserial –gv /dev/ttyS<*port number*> **/dev/ttyS4, UART: 16550A, Port: 0xdff0, IRQ: 11** If the port listing does not match the one found by lspci, reconfigure the device by entering the following: **linux#** setserial /dev/ttyS<*port number*> uart 16550a port <*port*> irq <*irq*> ^fourport

Common Questions

What do I do if intenable does not work immediately or causes a segmentation fault?

Enter the following to recompile interrupt_enable.c and rerun intenable. Also, the source code for intenable is available for editing and viewing at interrupt_enable.c provided in the PCI-SERIAL directory.

linux PCI-SERIAL# gcc interrupt_enable.c –o intenable
linux PCI-SERIAL# ./intenable <*PCI memory address found in lspci*>

What do I do if `FIFOtrigger` does not work immediately or causes a segmentation fault?

Enter the following to recompile `FIFOtrigger_pci.c` and rerun `FIFOtrigger`. Also, the source code for `FIFOtrigger` is available for editing and viewing at `FIFOtrigger_pci.c` provided in the `PCI-SERIAL` directory.

```
linux PCI-SERIAL# gcc -O FIFOtrigger_pci.c -o
                  FIFOtrigger
linux PCI-SERIAL# ./FIFOtrigger <port number>
                  <rx_trigger> <tx_trigger>
```

What do I do if `rs485` does not work immediately?

Enter the following to recompile `rs485.c`. Also, the source code for `rs485` is available for editing and viewing at `rs385.c`, provided in the `PCI-SERIAL` directory.

```
linux PCI-SERIAL# gcc -O rs485.c -o rs485
```

What do I do if `clockspeed` does not work immediately or if it causes a segmentation fault?

Enter the following to recompile `clock_speed.c` and rerun `clockspeed`. Also, the source code for `clockspeed` is available for editing and viewing at `clock_speed.c`, provided in the `PCI-SERIAL` directory.

```
linux PCI-SERIAL# gcc -O clock_speed.c -o clockspeed
linux PCI-SERIAL# ./clockspeed <port number> <high/low>
```

What do I do if `serialtest` does not work immediately or if it causes a segmentation fault?

Enter the following to recompile `serialtest.c` and rerun `serialtest`. Also, the source code for `serialtest` is available for editing and viewing at `serialtest.c`, provided in the `PCI-SERIAL` directory.

```
linux PCI-SERIAL# gcc serialtest.c -o serialtest
linux PCI-SERIAL# ./serialtest <receive port number>
                  <transmit port number>
```

What do I do if `serialtest` hangs?

Make sure the interface is seated correctly and tighten the screw that holds the interface in place. Also, make sure the cables are attached to the correct ports. In some cases, `serialtest` hangs if the transceiver modes (PCI-485) are not set. Try setting both transmit and receive ports to transceiver mode 0 (Four-Wire Mode).

```
linux# ./rs485 <transmit port number> 0
linux# ./rs485 <receive port number> 0
linux# ./serialtest <receive port number> <transmit port
       number>
```

What do I do if my ports are not communicating correctly and print strange characters?

Make sure the baud rate, character size, clock speed (if PCI-485) and parity are the same for both the receiver and transmitter. Also make sure the transceiver modes (PCI-485) have been selected for both transceivers.

How can I use `/dev/ttyS0`, `/dev/ttyS1`, `/dev/ttyS2`, or `/dev/ttyS3` as National Instruments serial ports?

Check for available serial devices by entering the following:

```
linux#  setserial -gv /dev/ttyS*
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
/dev/ttyS1, UART: unknown, Port: 0x02f8, IRQ: 3
/dev/ttyS2, UART: unknown, Port: 0x03e8, IRQ: 4
/dev/ttyS3, UART: unknown, Port: 0x02e8, IRQ: 3
```

Devices labeled with `UART: unknown` are available for use. To designate the available device when using `setserial`, enter the following:

```
linux#  setserial /dev/ttyS<port number> uart 16550a
        port 0x<port address> irq <irq> ^fourport
```

Assign Driver

Use these instructions to automatically assign the driver at startup using `/etc/rc.d/rc.serial`.

You should not change the `/etc/rc.d/rc.serial` file until you have your serial interfaces installed and configured. If you add another device or interface to your computer, and your `lspci` output changes, make sure you also change `/etc/rc.d/rc.serial`.

Depending on your Linux distribution, you might not have an `/etc/rc.d/` directory. In this case, your `rc.serial` would be located under `/etc/rc.serial`. If you do not have `/etc/rc.d/`, replace the references to `/etc/rc.d/` with `/etc/`. If you are using a Debian distribution, replace the references to `/etc/rc.d/rc.serial` with `/etc/rc.boot/0setserial`.

For more information about using `setserial`, refer to `linux # more /usr/doc/setserial*/README`.

1. Enter the following:

   ```
   linux#   cp /usr/doc/setserial*/rc.serial /etc/rc.d/
   linux#   pico /etc/rc.d/rc.serial
   ```

   a. Make sure that `SETSERIAL=` points to the correct location. Check your `/bin` and `/sbin` directories, and change the `SETSERIAL=` line to say either `SETSERIAL=/bin/setserial` or `SETSERIAL=/sbin/setserial`

   b. Under `AUTOMATIC CONFIGURATION`, leave the following lines uncommented and delete or comment out (by adding a # to the beginning of the line) all the other lines in the section. (Depending on your version of `setserial`, `ttyS` might replace `cua`):

   ```
   AUTO_IRQ=auto_irq

   ${SETSERIAL} /dev/cua0 ${AUTO_IRQ} skip_test
   autoconfig ${STD_FLAGS}

   ${SETSERIAL} /dev/cua1 ${AUTO_IRQ} skip_test
   autoconfig ${STD_FLAGS}

   ${SETSERIAL} /dev/cua2 ${AUTO_IRQ} skip_test
   autoconfig ${STD_FLAGS}

   ${SETSERIAL} /dev/cua3 ${AUTO_IRQ} autoconfig
   ${STD_FLAGS}
   ```

   c. Under `MANUAL CONFIGURATION`, comment out or delete everything but the following lines. Change the lines concerning `/dev/cua4-/dev/cua(4+`*`n`*`)` (where *n* is the number of ports on the PCI serial interface) to the following:

   ```
   # These are the first set of AST Fourport ports
   #
   ${SETSERIAL} /dev/ttyS4 uart 16550A port <port
   address> irq <irq> ^fourport
   ${SETSERIAL} /dev/ttyS5 uart 16550A port <port
   address> irq <irq> ^fourport
   ${SETSERIAL} /dev/ttyS6 uart 16550A port <port
   address> irq <irq> ^fourport
   ${SETSERIAL} /dev/ttyS7 uart 16550A port <port
   address> irq <irq> ^fourport
   ```

   Refer to the section *Sample /etc/rc.d/rc.serial File* for an example (configured according to the `lspci` example above).

2. Enter the following:

   ```
   linux#   pico /etc/rc.d/rc
   ```

3. Add the following segment to the end of the file (not necessary for Debian distribution):

```
if [ -f /etc/rc.d/rc.serial ]; then
    sh /etc/rc.d/rc.serial
    fi
```

Sample /etc/rc.d/rc.serial File

```
#
# /etc/rc.d/rc.serial
#       Initializes the serial ports on your system
#
# Distributed with setserial version 2.14
#
# Standard flags you want your serial devices to have
# Examples: SAK, pgrp_lockout, session_lockout
#
STD_FLAGS="session_lockout"
SETSERIAL=/bin/setserial
echo -n "Configuring serial ports...."
# Do wild interrupt detection
#
${SETSERIAL} -W /dev/cua0
################################################################
#
# AUTOMATIC CONFIGURATION
#
################################################################
# Do AUTOMATIC_IRQ probing
#
AUTO_IRQ=auto_irq
# These are the standard COM1 through COM4 devices
#
#
${SETSERIAL} /dev/cua0 ${AUTO_IRQ} skip_test autoconfig ${STD_FLAGS}
${SETSERIAL} /dev/cua1 ${AUTO_IRQ} skip_test autoconfig ${STD_FLAGS}
${SETSERIAL} /dev/cua2 ${AUTO_IRQ} skip_test autoconfig ${STD_FLAGS}
${SETSERIAL} /dev/cua3 ${AUTO_IRQ} autoconfig ${STD_FLAGS}
################################################################
#
# MANUAL CONFIGURATION
#
################################################################
```

```
# Changed for the two-port PCI-SERIAL interface.
#
${SETSERIAL} /dev/ttyS4 uart 16550A port 0xdff0 irq 11 ^fourport
${SETSERIAL} /dev/ttyS5 uart 16550A port 0xdfe0 irq 11 ^fourport
############################################################
#
# Print the results of the serial configuration process
#
############################################################
echo "done."
${SETSERIAL} -bg /dev/cua? /dev/cua??
```

322539A-01 Aug99