USING AT 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 AT-232/2, AT -232/4, AT-485/2, AT-485/4, AT-232/2 Isolated, AT-232/4 Isolated, AT-485/2 Isolated, and AT-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 manual:

- *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 AT 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 Plug and Play support and the following options for Character Devices when you configure and recompile the kernel using `make menuconfig`.

  – Standard/generic dumb serial support

  – Extended dumb serial driver options

  – Support more than four serial ports

  – Support for sharing serial interrupts

- `isapnptools` 1.17 or later. To find the version number of `isapnptools`, enter the following:

  `linux#` `isapnp`

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

  ```
  linux# setserial -V
  ```

- AT-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 AT-SERIAL.tar.gz
  ```

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

  ```
  linux#                cd AT-SERIAL
  linux AT-SERIAL#      ls
     FIFOtrigger        rs485    serialtest
     FIFOtrigger_at.c   rs485.c  serialtest.c
     termios_program.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 isapnp 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 and configure the port address and IRQ of your AT serial interface.

   a. Enter the following:

   ```
   linux# pnpdump > isafile
   linux# pico isafile
   ```

b. Under the National Instruments serial device, uncomment the following lines (by removing the # sign in front of the line):

```
(IO 0 (SIZE 8)(BASE <port address>))
(IO 1 (SIZE 8) (BASE <port address>))
(INT 0 (IRQ <irq> (MODE +E)))
(ACT Y)
```

- When you uncomment the lines, make sure that each serial port on the interface has a different base address (<*port address*>) and that you have as many `(IO ? (SIZE 8)(BASE <port address>))` entries as you have serial ports.
- Remove the `(CHECK)` if it appears in any of the uncommented lines.

c. Enter the following:

```
linux# isapnp isafile
```

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

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

4. If you have an AT-485 interface, set the transceiver mode for each serial port. Refer to the section *Select AT-485 Transceiver Mode* for more information about selecting an AT-485 transceiver mode.
5. After you connect a cable between the two ports, test the setup by running `serialtest` (from the `AT-SERIAL` directory).

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

Setup

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

Configure Interface

Follow these instructions to find and configure the port address and IRQ of your AT serial interface.

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

1. Enter the following:

   ```
   linux# pnpdump > isafile
   linux# pico isafile
   ```

   You can use any editor you are comfortable with. Under the National Instruments serial device, uncomment the following lines (by removing the # sign in front of the line):

   ```
   (IO 0 (SIZE 8)(BASE <port address>))
   (IO 1 (SIZE 8)(BASE <port address>))
   (INT 0 (IRQ <irq> (MODE +E)))
   (ACT Y)
   ```

   - When you uncomment the lines, make sure that each serial port on the interface has a different base address (<*port address*>) and that you have as many (IO ? (SIZE 8)(BASE <*port address*>)) entries as you have serial ports. To determine the <*port address*>, use the listed value of <*port address*> for the first port. Increment each of the following ports by 16 (0x10 in hex).
   - Remove the (CHECK) in any of the uncommented lines.
   - If you encounter problems using the <*irq*> supplied in the isafile, refer to the setserial man page for more IRQ information.

2. Enter the following:

```
linux# isapnp isafile
```

Note Repeat Step 2 each time you restart your computer.

isafile Example

The edited `isafile` file for a two-port AT serial interface should be similar to following (refer to *Sample isafile after Edit* for a sample `isafile`). If you have an old AT interface that does not share interrupts, refer to *Sample isafile after Edit for Non-Interrupt Sharing Interfaces*.

```
# (DEBUG)
(READPORT 0x???)
(ISOLATE PRESERVE)
(IDENTIFY *)
(VERBOSITY 2)
(CONFLICT (IO FATAL)(IRQ FATAL)(DMA FATAL)(MEM FATAL))
# information about other interfaces...
(CONFIGURE NICd???/???????? (LD ?
(IO 0 (SIZE 8)(BASE 0x0100)) # <port address> = 0x100
(IO 1 (SIZE 8)(BASE 0x0110)) # <port address> = 0x110
(INT 0 (IRQ 3 (MODE +E))) # <irq> = 3
(ACT Y)
))
```

Assign Serial Driver

Assign the serial driver to your devices.

Note Repeat this step each time you restart your computer or until you set up your `/etc/rc.d/rc.serial` file. (Refer to *Assign Driver* 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 the `<port address>` and `<irq>` values from your edited `isafile`.

```
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 address can lock up your machine.

setserial Example

Enter the following to assign the serial driver to your devices for the values in the above *isafile Example*:

```
linux#   setserial /dev/ttyS4 uart 16550A port 0x100 irq
         3 ^fourport
linux#   setserial /dev/ttyS6 uart 16550A port 0x110 irq
         3 ^fourport
```

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*
```

For the `isafile` 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: 0x0100, IRQ: 3
/dev/ttyS5, UART: 16550A, Port: 0x0110, IRQ: 3
```

Enable FIFO Buffers

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

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

Table 1-1 shows the receive FIFO trigger levels and corresponding `rx_trigger` values.

Table 1-1. rx_trigger Values

Receive FIFO Trigger Level	rx_trigger
1	0x00
4	0x40
8	0x80
14	0xC0

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 the FIFO buffers, refer to your serial getting started manual.

FIFO Example

Enter the following to set the receive FIFO trigger level to 14 for `/dev/ttyS5`:

```
linux AT-SERIAL# ./FIFOtrigger 5 0xC0
```

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 each serial port. 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 `AT-SERIAL` directory.

Select AT-485 Transceiver Mode

If you are using an AT-485 interface, you can select the transceiver mode for each device. For more information on the transceiver control modes, refer to your serial getting started manual. Use the `rs485` program (from your `AT-SERIAL` directory) to select the transceiver mode. `rs485` sets the transceiver mode for only one serial port. To set the transceiver mode for your other serial ports, rerun `rs485` with a different serial port specified in the command line. If you do not know which transceiver mode to use, choose Four-Wire Mode.

Table 1-2. Transceiver Control Modes

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

rs485 Example

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

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

Test the Setup

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

```
linux AT-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 and enter the following:

```
linux AT-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	irs485:**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 configured by the isapnp step, reconfigure the device by entering the following: **linux#** setserial /dev/ttyS<*port number*> uart 16550a port <*port address*> irq <*irq*> ^fourport

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

Recompile `FIFOtrigger_at.c` and rerun `FIFOtrigger` by entering the following. Also, the source code, `FIFOtrigger_at.c`, is available for viewing and editing.

```
linux AT-SERIAL# gcc -O FIFOtrigger_at.c -o FIFOtrigger
linux AT-SERIAL#./FIFOtrigger <port number>
                 <rx_trigger>
```

What do I do if `rs485` does not work immediately or if it causes a segmentation fault?

Recompile `rs485.c` and enter the following to rerun `rs485`. Also, the source code, `rs485.c`, is available for viewing and editing.

```
linux AT-SERIAL# gcc -O rs485.c -o rs485
linux AT-SERIAL# ./rs485 <port number> <mode>
```

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

Recompile `serialtest.c` and enter the following to rerun `serialtest`. Also, the source code, `serialtest.c`, is available for viewing and editing.

```
linux AT-SERIAL# gcc serialtest.c -o serialtest
linux AT-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 (AT-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, and parity are the same for both the receiver and transmitter. Also, make sure the transceiver modes (AT-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 `isafile` 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` could 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. You need to change the lines concerning /dev/cua4-/dev/cua(4+*n*) (where *n* is the number of ports on your serial interface) to the following:

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

Refer to the section *Sample /etc/rc.d/rc.serial File* for an example (configured according to the *Sample isafile after Edit*).

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 distributions):

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

Sample isafile after Edit

```
# $Id: pnpdump.c,v 1.18 1999/02/14 22:47:18 fox Exp $
# This is free software, see the sources for details.
# This software has NO WARRANTY, use at your OWN RISK
#
# For details of this file format, see isapnp.conf(5)
#
# For latest information and FAQ on isapnp and pnpdump see:
# http://www.roestock.demon.co.uk/isapnptools/
#
# Compiler flags: -DREALTIME -DNEEDSETSCHEDULER -DABORT_ONRESERR
#
# Trying port address 0203
# Board 1 has serial identifier f2 00 a5 8a 6b 21 d1 23 39
# (DEBUG)
(READPORT 0x0203)
(ISOLATE PRESERVE)
(IDENTIFY *)
(VERBOSITY 2)
(CONFLICT (IO FATAL)(IRQ FATAL)(DMA FATAL)(MEM FATAL)) # or WARNING
# Card 1: (serial identifier f2 00 a5 8a 6b 21 d1 23 39)
# Vendor Id NICd121, Serial Number 10848875, checksum 0xF2.
# Version 1.0, Vendor version 0.0
# ANSI string -->National Instruments AT-485/2 IRQ Share (PnP)<--
#
# Logical device id NICd121
#     Device support I/O range check register
#     Device supports vendor reserved register @ 0x38
#     Device supports vendor reserved register @ 0x3a
#
# Edit the entries below to uncomment out the configuration required.
# Note that only the first value of any range is given, this may be
# changed if required
# Don't forget to uncomment the activate (ACT Y) when happy
(CONFIGURE NICd121/10848875 (LD 0
#     Logical device decodes 16 bit IO address lines
#         Minimum IO base address 0x0100
#         Maximum IO base address 0x03f8
#         IO base alignment 8 bytes
#         Number of IO addresses required: 8
(IO 0 (SIZE 8) (BASE 0x0100))
#     Logical device decodes 16 bit IO address lines
#         Minimum IO base address 0x0100
#         Maximum IO base address 0x03f8
```

```
#         IO base alignment 8 bytes
#         Number of IO addresses required: 8
(IO 1 (SIZE 8) (BASE 0x0110))
#     IRQ 3, 4, 5, 7, 10, 11, 12 or 15.
#         High true, edge sensitive interrupt (by default)
(INT 0 (IRQ 3 (MODE +E)))
(NAME "NICd121/10848875[0]{National Instruments AT-485/2 IRQ Share
(PnP)}")
(ACT Y)
))
# End tag... Checksum 0x00 (OK)
# Returns all cards to the "Wait for Key" state
(WAITFORKEY)
```

Sample isafile after Edit for Non-Interrupt Sharing Interfaces

A non-interrupt sharing interface will have a logical device entry for each port on the interface. Also, you need to have a free IRQ for each port.

```
# (DEBUG)
(READPORT 0x0203)
(ISOLATE PRESERVE)
(IDENTIFY *)
(VERBOSITY 2)
(CONFLICT (IO FATAL)(IRQ FATAL)(DMA FATAL)(MEM FATAL)) # or WARNING
# Card 1: (serial identifier 36 00 a3 57 f1 01 d3 23 39)
# Vendor Id NICd301, Serial Number 10704881, checksum 0x36.
# Version 1.0, Vendor version 0.1
# ANSI string -->National Instruments AT-485/4 (Plug and Play)<--
#
# Logical device id NICd301
#     Device support I/O range check register
#     Device supports vendor reserved register @ 0x38
#     Device supports vendor reserved register @ 0x3a
#
(CONFIGURE NICd301/10704881 (LD 0
(IO 0 (SIZE 8) (BASE 0x0100))
(INT 0 (IRQ 3 (MODE +E)))
(NAME "NICd301/10704881[0]{National Instruments AT-485/4 (Plug and
Play)}")
(ACT Y)
))
#
# Logical device id NICd301
```

```
#     Device support I/O range check register
#     Device supports vendor reserved register @ 0x38
#     Device supports vendor reserved register @ 0x3a
#
(CONFIGURE NICd301/10704881 (LD 1
(IO 0 (SIZE 8) (BASE 0x0110))
(INT 0 (IRQ 4 (MODE +E)))
 (NAME "NICd301/10704881[1]{National Instruments AT-485/4 (Plug and
Play)}")
(ACT Y)
))
#
# Logical device id NICd301
#     Device support I/O range check register
#     Device supports vendor reserved register @ 0x38
#     Device supports vendor reserved register @ 0x3a
#
(CONFIGURE NICd301/10704881 (LD 2
(IO 0 (SIZE 8) (BASE 0x0120))
(INT 0 (IRQ 5 (MODE +E)))
(NAME "NICd301/10704881[2]{National Instruments AT-485/4 (Plug and
Play)}")
(ACT Y)
))
#
# Logical device id NICd301
#     Device support I/O range check register
#     Device supports vendor reserved register @ 0x38
#     Device supports vendor reserved register @ 0x3a
#
(CONFIGURE NICd301/10704881 (LD 3
(IO 0 (SIZE 8) (BASE 0x0130))
(INT 0 (IRQ 10 (MODE +E)))
(NAME "NICd301/10704881[3]{National Instruments AT-485/4 (Plug and
Play)}")
(ACT Y)
))
(WAITFORKEY)
```

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 AT-SERIAL interface.
#
${SETSERIAL} /dev/ttyS4 uart 16550A port 0x100 irq 3 ^fourport
${SETSERIAL} /dev/ttyS5 uart 16550A port 0x110 irq 3 ^fourport
```

```
############################################################
#
# Print the results of the serial configuration process
#
############################################################
echo "done."
${SETSERIAL} -bg /dev/cua? /dev/cua??
```

322538A-01 Aug99