

ft_serial

This chapter describes the functionality of `ft_serial` in providing serial connection to the Netra t 1800 system.

The chapter has the following sections:

- “File System Access to the Console” on page 28
- “Configuring Serial Connections Using the CMS” on page 33

Accessing the Console

The Netra ft 1800 console can be accessed from multiple physical ports. In the default configuration, the console port on both CAFs provide access to the console. The console port on side A CAF is connected to a separate serial communications controller located on the side A motherboard. Similarly, the console port on the side B CAF is connected to a serial controller on the B motherboard.

This configuration guarantees console availability:

- During motherboard hot swap
- During CAF replacement
- After the failure of one of the serial controllers

During split-mode operation, each serial controller is allocated to its natural side. The consequence of this allocation is that fault-tolerant serial connections via the CAFs are not available when the system is split.

The serial controllers used on the Netra ft 1800 system provide two independent serial channels, with the input and output lines from the first channel connected to the CAF console port, and the lines from the second channel connected to the modem port on the same CAF.

To ensure the console configuration is fault tolerant, only ports on separate CAFs should be combined.

The methods by which different ports can be configured to provide console access are:

- Through the Configuration Management System (CMS)—the preferred method
- By using the `ttymuxadm (1M)` utility

File System Access to the Console

Fault-tolerant serial connections are provided by a separate STREAMS multiplexor driver called `ttymux` (alias name is `u4ftser`). To support booting, OpenBoot PROM creates a device node corresponding to the multiplexor with the path `/u4ftser@0,0`. The minor devices created by the serial multiplexor appear in the Solaris devices directory as:

- `/devices/u4ftser@0,0:[con|ctl]`
- `/devices/u4ftser@0,0:sm[a-f],cu`
- `/devices/u4ftser@0,0:sm[a-f]`

The corresponding device links to these files are:

- `/dev/ttymux:[con|ctl]`
- `/dev/term/[0-5]`
- `/dev/cua/[0-5]`

Each of these multiplexor devices can be associated with one or more real serial devices. If a real serial device is associated with a multiplexor device, real input and output can be performed by opening the multiplexor device. When two or more real serial devices on separate chips are associated with a multiplexor device, the multiplexor provides fault-tolerant service—any output is sent to all associated devices, and any input is multiplexed onto a single input stream and made available for reading.

Real serial devices are associated in a two-stage process:

1. Link the device underneath the `ttymux` driver
2. Associate the linked device with a multiplexor device

Both steps can be performed together using the `ttymuxadm` utility. Otherwise, a program must be written that issues a STREAMS link ioctl (`I_PLINK`), and then issues a `ttymux` driver-specific ioctl (`TTYMUX_ASSOC`).

A linked lower device cannot be accessed directly. Although you can open such a device, any reads or writes will fail.

Reading the Current Multiplexor Configuration

All linked devices and associations can be read with the `ttymuxadm` utility (using the `-i` option), or by issuing a driver-specific `ioctl` (`TTYMUX_LIST`). The `ttymuxadm -i` command provides configuration information in the following format:

```
A-CAF:1 20:1 /dev/term/b 31 (166:5 1095586644)
```

The first four fields provide information about a linked device:

- FRU name and port number
- Device number of the linked device
- Device path of the linked device
- STREAMS link identifier for the link

The information in brackets indicates whether the linked device is associated with one of the multiplexor devices (the two fields correspond to the device number of the associated multiplexor device, followed by a user-supplied identifier for the association, which is required when the association is destroyed).

If the lower device number is reported as `-1:-1`, the lower device is not linked. If the multiplexor device number is `-1:-1`, the linked lower device is not associated with any of the multiplexor devices.

Files

TABLE 2-1 Files Used by `ft_serial`

File	Function
<code>/kernel/drv/ttymux.conf</code>	Driver configuration file
<code>/etc/init.d/ttymuxrc</code>	Run at rcS to configure any persistent ttymux associations
<code>/sbin/ttymuxadm</code>	Serial multiplexor administration utility; uses driver-specific <code>ioctl(2)</code>
<code>/dev/term/[0-5]</code>	Links to the <code>/devices</code> directory for opening serial lines with dial-in semantics
<code>/dev/cua/[0-5]</code>	Links to the <code>/devices</code> directory for opening serial lines with dial-out semantics
<code>/dev/ttymux:con</code>	Link to the <code>/devices</code> directory for opening a fault-tolerant console
<code>/dev/ttymux:ctl</code>	Link to the <code>/devices</code> directory for issuing driver-specific <code>ioctls</code> . A specific minor device is required since all other devices may be in use.

Boot Time Configuration of the Console

If the OpenBoot PROM environment variables `input-device` and `output-device` are set to the `u4ftser` node in the PROM, UNIX attempts to configure fault-tolerant consoles during boot. Any failure during boot is written to the system console.

Three message severities are issued:

- Information
- Notices
- Warnings

Warnings should be regarded as serious events that require investigation. The only *normal* message to be expected occurs when a serial device is inaccessible because the hardware is offline, broken or not owned.

In the following, `%d` is a placeholder for a UNIX error code (see `intro(2)`), and `%s` represents a device path.

```
Information: rconsconfig: TCSETS %d
```

Whilst a serial device was being plumbed (configured) underneath the multiplexor, the boot code was unable to enable the receiver on the serial device. The consequence is that input will not be taken from that device until a successful `ioctl` is issued (any `ioctl` that modifies the terminal input or control flags, enables the receiver). When the login monitor for the console is configured, the receiver is enabled.

```
Notice: rconsconfig: TCGETS %d
```

Unable to read the `termios` control word from a serial device driver. This error causes the receiver to be left disabled, as in the previous error message.

```
Information: rconsconfig: associate error %d
```

The `ttymux` driver has been unable to associate a plumbed device with the console STREAM. The precise reason for the failure is reported in the system status log (look for lines labeled `ttymux`). If no consoles can be plumbed (configured), the system will still boot. Investigatory and remedial action can be taken when network logins have been enabled.

```
Information: rconsconfig: prop update failed
```

An update of the `console-devices` property of the multiplexor has failed. The impact of this failure is that both software and hardware aborting to OpenBoot PROM will not work until the redirection device (`cn driver`) is opened.

```
Information: rconsconfig: no dev info for mux
```

There is no device node for the multiplexor. The impact is the same as for the previous message.

```
Information: rconsconfig: device at %s is inaccessible
```

A target console device is inaccessible. This can happen if:

- There is no hardware at the path given in the %s string
- The hardware is unusable (broken or disabled by the OpenBoot PROM, or in use by the other half of a split system)

```
Information: rconsconfig: bad open on %s
```

A target console cannot be opened. Typically, this can occur if the hardware is broken, or if the hardware fails while the open is initializing the hardware.

```
Information: rconsconfig: bad mux link on %s
```

The `ttymux` driver has been unable to plumb a serial device. The precise reason for the failure is reported in the system status log (look for lines labeled `ttymux`). If no consoles can be plumbed (configured), the system will still boot. Investigatory and remedial action can be taken when network logins have been enabled.

```
Information: rconsconfig: bad close on %s
```

After successfully linking a console device, that device could not be closed. This can result in attempts to take the serial hardware offline to fail because UNIX assumes the device is still in use. Failure to close a device is symptomatic of a software bug and, as such, this message should never be seen.

```
Notice: rconsconfig: invalid console-flags - using 0x%x
```

A corrupt set of console flags has been read, so a default set has been chosen. The only standard way to repair them is from the OpenBoot PROM prompt.

```
Warning: rconsconfig: bad prop lookup
```

The property that specifies which consoles to multiplex, does not specify any consoles. The system still boots, but with no console access.

```
Warning: rconsconfig: autopush not set
```

Unable to configure the autopush(1M) facility on the console. This message means that the `ttycompat` and `ldterm` modules will not be pushed onto the console stream when it is opened.

```
Warning: rconsconfig: open %d failed (no console)
```

Unable to open the system console device. This message is generally a result of setting the OpenBoot PROM *input-device* and *output-device* variables to different or invalid paths. If the device corresponding to a given path is inaccessible, the path is considered to be invalid. If a valid console device cannot be found, the system will panic when the redirection console (cn driver) is opened. You must ensure that *input-device* and *output-device* reference a valid console.

```
Warning: rconsconfig: no real consoles plumbed
```

Unable to multiplex any real consoles. This error is reported in conjunction with one or more of the previous messages. It can occur if the list of consoles to multiplex is empty, or if all of the target consoles cannot be plumbed.

Configuring Serial Connections Using the CMS

Redundant (or fault-tolerant) serial connections can be configured using the CMS or the `ttymuxadm(1M)` utility. In fact, the CMS provides a front end to `ttymuxadm`. However, the CMS is the preferred method because it provides information about required devices, such as CAFs, serial chips and motherboards, and it integrates serial functionality into the overall management system.

▼ To Configure a Nonredundant Serial Connection

1. Determine which file system entry in `/dev/term` will provide access to the new serial connection.

2. Type:

```
cmsconfig
```

If you have chosen `/dev/term/0`, the instance number 1 of `ft_serial` is displayed (`/dev/term/1` corresponds to `ft_serial 2`, and so on).

3. Type the following command to include it in the display:

```
i ft_serial 1
```

4. Press return.

An entry for `ft_serial 1` should be displayed. Enter its Item number. This will display a screen containing a list of the attributes that are relevant to configuring `ft_serial` serial objects.

For example:

```
Select: ft_serial 1
Item   Name                               Value                               Page 1 of 1
-----
0      state                               offline
1      description                          serial multiplexer
2      user_label
3      port0                                NULL
4      port1                                NULL
5      port2                                NULL
6      port3                                NULL
7      info
8      redundancy                           not_present
9      mode_of_use                           /dev/term/0
10     ports_in_use
11     working_ports
12     busylock                             no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

Note – If the `ft_serial` object is in the `not_present` state, it must to be configured.

5. The four port fields correspond to the four possible physical CAF ports. To configure a nonredundant port, select the required CAF port:

- port 0 is the console port on the A-CAF
- port 1 is the modem port on the A-CAF
- port 2 is the console port on the B-CAF
- port 3 is the modem port on the B-CAF

For example, selecting Item 4 displays the following screen;

```
Modify: ft_serial 1 port1
Item   Value
-----
0      NULL
1      A-CAF_Modem
Page 1 of 1
(H)elp, <Number> to set value or (Q)uit ?
```

6. Enter 1 to configure a nonredundant serial connection (to the modem port on the A CAF) on `/dev/term/0`.

If the operation is successful, the `redundancy` attribute changes to `non_redundant` as follows:

```

Select: ft_serial 1
Item   Name                Value                Page 1 of 1
-----
0      state                 online
1      description            serial multiplexer
2      user_label
3      port0                  NULL
4      port1                  A-CAF_Modem
5      port2                  NULL
6      port3                  NULL
7      info
8      redundancy            non_redundant
9      mode_of_use            /dev/term/0
10     ports_in_use           A-CAF_Modem
11     working_ports          A-CAF_Modem
12     busylock                no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?

```

If the operation is unsuccessful (for instance, through trying to configure port2 on the A side of a split system), the redundancy attribute remains as not_present, and the info attribute indicates why the redundancy and state attributes have not changed:

```

Select: ft_serial 1
Item   Name                Value                Page 1 of 1
-----
0      state                 offline
1      description            serial multiplexer
2      user_label
3      port0                  NULL
4      port1                  NULL
5      port2                  NULL
6      port3                  B-CAF_Modem
7      info                   3 (unusable)
8      redundancy            not_present
9      mode_of_use            /dev/term/0
10     ports_in_use
11     working_ports
12     busylock                no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?

```

If the redundancy level (`not_present`, `non_redundant`, `dual_redundant`, and so on) does not correspond to the required number of configured ports the `info` attribute indicates which ports have failed to configure, and the reason why.

In the above example, port 3 has failed to configure because it is unusable. This is an indication that the associated port object is unusable (for example, its associated CAF may not be enabled).

Configuring a Redundant Serial Connection

Repeat Step 5 of the previous example by choosing a different port, such as port 2. If the operation succeeds, the `ft_serial` object is set to `dual_redundant`.

```
Select: ft_serial 1
Item  Name                Value                Page 1 of 1
-----
0     state                 online
1     description            serial multiplexer
2     user_label
3     port0                  NULL
4     port1                  A-CAF_Modem
5     port2                  NULL
6     port3                  B-CAF_Modem
7     info
8     redundancy             dual_redundant
9     mode_of_use            /dev/term/0
10    ports_in_use           A-CAF_Modem B-CAF_Modem
11    working_ports         A-CAF_Modem B-CAF_Modem
12    busylock               no

(H)elp, <Number> to modify, (S)elect, (T)op or (Q)uit ?
```

Successfully configuring other ports increases the redundancy level still further.

Configuring Highly Redundant Serial Connections

The maximal level of redundancy that can be configured using the CMS is `quad_redundant`, where all 4 CAF ports have been configured.

The number of devices that can be configured is controlled by one of the `ttymux` driver properties. Further devices can be configured into the redundant set using `ttymuxadm`. For example, if a Serial Asynchronous Interface PCI card is plugged into one of the PCI slots, port entries `/dev/term/a00[0-7]` can be configured into the redundant set using the command:

```
ttymuxadm -a /dev/term/a000 /dev/term/0
```

If more than four ports are configured in this way, the CMS displays the redundancy as highly redundant. However, only CAF ports show as being configured.

Setting Terminal Characteristics on Redundant Serial Devices

Terminal characteristics such as speed, parity, character sizes flow control, and so on, must be set consistently at both ends of a serial connection. Therefore, the terminal settings on the remote terminal device must match the settings on the port to which it is attached. When redundant ports are multiplexed, the only way to modify terminal settings is by means of the multiplexor stream associated with the underlying devices, and the settings will be applied to all connections. If the remote devices do not have the same settings, one of the pair will produce garbled I/O.

In addition, since input streams from redundant ports are interleaved onto a single stream, only character-orientated protocols can run over a redundant serial connection. Message boundaries are not respected; only raw character I/O is supported. If data messages are encoded in the input stream, they are likely to become corrupted with data received on the other input streams.

For example, typing:

```
# eval ` /usr/openwin/bin/resize `
```

on a redundant console device fails if all the terminal devices are connected, and responds by returning their current window size.