Release Note

RIO for LynxOS v2.5 & v3.0

Release v1.2

5600034-17 November 2001

1. Introduction

This is an engineering release note to accompany release v1.2 of the RIO Device Driver for the Lynx Real-Time Operating System. It is not intended to be a complete and exhaustive User Guide, but describes significant changes to and limitations of the latest release and gives instructions on how to install and uninstall the driver.

The following file is installed as part of the driver's fileset for future reference :

/rio/README	-	engineering description of current release, includes basic description of changes.

Even if you are familiar with previous releases of this driver, *please* take a few moments to read this release note and follow any instructions given - It may well save you some time.

Contents

2. Scope

This release note is applicable only to Driver and O/S versions as detailed below unless advised otherwise :

- RIO Device Driver for LynxOS
  Release v1.2.2
  Release date : 23 December, 1999

- Associated fileset archive : `rio122.tar`

- Compatible with O/S versions v2.5.x & v3.0.x

3. Changes in this release

The latest release is v1.2.2, and includes :

- Text based configuration tools in pace of config.rio.
- Support for Dual-Host Fail-Safe operation.
- Driver Statistics reporting.

Release 1.2.2 is a bug fix release and has correct support for ioctls which facilitate control and monitoring of modem signals. The ioctls are described in detail at Appendix C. Please refer to the README file installed with driver for bug fix details.

Following the introduction of the text based configuration tools, config.rio and the old rioadopt have been removed from the driver's installation fileset. The new, text based, configuration tools provide a new version of rioadopt, which has significantly different functionality to the previuos version.

You MUST ensure that config.rio and the previous version of rioadopt are not used following installation of this release. Ideally the old configuration tools should be removed from the system, preferably by following the recommendations given at the beginning of the section headed "Driver Installation".

Please refer to Appendix D for full details of the new configuration tools, (statistics reporting is also covered here - see the 'riostats' utility).

Appendix E gives an example of how the set up Dual-Host Fail-Safe and undertake a Master/Slave swap.

4. Limitations/Concessions

There are no known major limitations.

5. Driver Installation

Note : This release uses a significantly different installation method to some previous releases. It is, therefore, strongly recommended this release is installed only on systems that have not previously had RIO installed, preferably use a clean O/S installation.

The driver fileset archive, (see section headed “Scope”), may be found on the CDROM supplied with this product or downloaded from the Perle web-site, (http://www.perle.com/). These instructions assume that the driver fileset archive, is resident in the root directory. It is also assumed that RIO hardware is installed and appropriately connected.

1. Ensure you are in the root directory ‘/’ with root privileges and ‘unpack’ the driver fileset onto the file-system :

   ```
   tar -xvf rio122.tar
   ```

   During ‘unpacking’ a list of installed files will be displayed.

   When ‘unpacking’ is complete you may, if you wish, delete the archive file to save disk space.

2. Run the ‘Install’ utility and respond appropriately to any prompts :

   ```
   Install.rio
   ```

   The install utility will display the following banner and prompt

   ```
   **** Specialix RIO Driver Installation/Configuration Utility ****

   Do you want to proceed (yY/nN) [Y]?
   ```

   This is your chance to abort the installation if you want to, otherwise press return.

   The install utility will detect that you have not previously installed RIO and display a brief message about host card support. Note that this release does not allow you to enable ‘polled’ operation.

   If a previous driver installation is detected you will be given prompts as detailed in the section headed “Driver Re-Installation”.

   The following message and prompt gives you the opportunity to configure ISA cards.

   ```
   You DO NOT have RIO ISA cards configured

   Add ISA Host Card configuration details (yY/nN) [N] ? y
   ```

   Press return if you are using PCI cards, an affirmative response results in the following prompts for ISA configuration information. Each prompt includes a default value enclosed in square brackets, press return to accept it.

Notes : A. You can only use the default values for ONE ISA host card.

B. Each host card requires 64k of address space. Adjacent cards must therefore be at least 0x10000 apart.

C. Start addresses must be aligned on a 32k (0x8000) boundary, (i.e. The last four hex digits must be ‘0000’ or ‘8000’).

CAUTION: **The user must determine that resources entered here are available and do not clash. No checking is done to ensure that addresses and IRQs entered by the user are either available or do not clash with previously defined RIO ISA cards.**

```
Base address for the card can be 0x10000 - 0xF00000
Enter base address in hex : [0xD0000]

The interrupt vector IRQ  could be 9, 11, 12 or 15
Enter the irq assignment : [9]
```

You will be prompted again to add further ISA cards. Press return when you have entered configuration details for each of the ISA cards you intend to use.

```
Add ISA Host Card configuration details (yY/nN) [N] ?
```

The installation will continue and the kernel will be re-linked.

3. When installation is complete, re-boot your system as instructed.
4. During system start-up you should see console messages displaying driver version, host card parameters and messages relating to established RTA connections.

6. Driver Re-Installation

If you want to change the host card configuration, install a new driver object library or just re-link RIO into the kernel (after an Uninstall), you should follow the instructions below.

1. Run 'Install.rio' as described in steps 1 & 2 of the section headed "Driver Installation".
2. The utility will recognise the fact that the RIO driver is already installed and display the following message, (after the initial banner and proceed prompt) :

   ```
   The Specialix RIO Driver is already installed in your system
   ```

   If you just want to re-link RIO into the kernel, press return at both the following prompts.

   You will be given the opportunity to let the utility know that you wish to change the host card configuration with the following prompt.

   ```
   Review and/or change Host Card configuration details (yY/nN) [N] ?
   ```

   You will be given the opportunity to let the utility know that you wish install a new driver object library with the following prompt.

   ```
   Re-install the driver object library (yY/nN) [N] ?
   ```

   Refer to the notes in the section headed "Driver Installation" before you proceed. If you provide an affirmative response to the first prompt and you already have ISA cards

configured, the address and IRQ details for each cards will be displayed together with a prompt to retain, change or delete as shown below.

```
You have an ISA card configured at Address 0xA0000 using IRQ 15
Select an option, <return> = r  -
    Retain   (r)
    Change   (c)
    Delete   (d)    [r] :
```

If you select 'change', you will be prompted to re-enter address and IRQ details as described in the section headed "Driver Installation", the current values will be the default. Press return to accept defaults, or enter new values as appropriate.

If you want to use PCI cards in place of configured ISA cards simply delete ISA card configuration details.

This process will repeat until all previously configured cards have been reviewed, the re-installation will then continue and the kernel will be re-linked.

3. When installation is complete, re-boot your system as instructed.
4. During system start-up you should see console messages displaying driver version, host card parameters and messages relating to established RTA connections.

7. Uninstalling the Driver

Note: Due to the way the driver is currently installed, it is not possible to easily remove the driver's fileset. If you wish to completely remove RIO driver files, simply get a listing of the files in the fileset archive, (see section headed "Scope" 2), and remove them manually. You may also wish to remove all RIO driver objects from the library files in /sys/lib, (please contact Customer Services should you require assistance).

The 'uninstall' procedure described below simply removes the RIO driver's configuration information from the Kernel configuration files (and driver objects from the kernel libraries if you wish) and carries out a Kernel re-build. The next time you re-boot, RIO hardware will be inactive.

1. Ensure you have root privileges and run the 'Uninstall' utility and respond appropriately to any prompts :

```
Uninstall.rio
```

The uninstall utility will display the following banner and prompt

```
***** Specialix RIO Driver de-installation Utility *****
```

```
Do you want to proceed (yY/nN) [Y]?
```

This is your chance to abort the de-installation if you want to, otherwise press return.

You will also be given the opportunity to uninstall RIO driver objects from the kernel library(ies). If you want to install a new driver library you do not have to do this first as the re-install will overwrite existing library objects.

```
Do you want to uninstall the driver objects (yY/nN) [N] ? y
```

2. When the uninstall is complete, re-boot your system.

Should you wish to re-activate RIO hardware and have not removed driver files, follow the re-installation procedure, (see the section headed “Driver Re-Installation”). Otherwise follow the installation procedure in its entirety.

Appendix A - ioctl – ‘RIO_QUICK_CHECK’

This appendix gives an example - program rioqc – of using the RIO_QUICK_CHECK ioctl.

```
/*
** ‘rioqc’ - Test operation of the 'RIO_QUICK_CHECK' ioctl
*/

#include <stdio.h>
#include <errno.h>
#include <fcntl.h>
#include <rio/rioioctl.h>

#define PERMS 0

char    *progName       = NULL,
        *rio_ctrl_dev   = "/dev/rio";

int     fd,
        ioctl_failed = 0,
        rio_rta_dis_cons;

main(int argc, char *argv[])
{
    /* see if we can open rio control device */
    if (( fd = open( rio_ctrl_dev, O_RDWR, PERMS )) == -1 )
    {
        perror("unable to open RIO control device");
        exit(1);
    }

    while ( !ioctl_failed )
    {
        if ( ioctl( fd, RIO_QUICK_CHECK, &rio_rta_dis_cons ) == -1 )
        {
            perror("RIO_QUICK_CHECK ioctl failed");
            ioctl_failed = 1;
            exit(1);
        }
        else
        {
            printf("RIO RTA Disconnections : %d\n", rio_rta_dis_cons );
        }
        sleep(1);
    }

    close(fd);
    exit(0);
}
```

Appendix B - RIO Hardware flow control

This appendix gives details of how to use hardware flow control with RIO.

LynxOS does not provide a hardware flow control mechanism via the stty command, so RIO provides 4 ioctls to facilitate it, these ioctls are defined in `/usr/include/rio/rioioctl.h`, which you will need to "#include" in your application if you want to use hardware flow control.

The ioctls have no arguments and are described below :

TCRIOCTSFLOWEN	- enable CTS flow control
TCRIOCTSFLOWDIS	- disable CTS flow control
TCRIORTSFLOWEN	- enable RTS flow control
TCRIORTSFLOWDIS	- disable RTS flow control

If you want to manipulate hardware flow control from the command line you will have to do this by writing a simple application.

An example using two of the ioctls to enable input and output flow control is given on the following page. Please note that it is not necessarily intended to be a working application, but a demonstration of how to use the RIO ioctls.

```
/* ioctltst.c */

#include <stdio.h>
#include <errno.h>
#include <fcntl.h>
#include <termio.h>
#include <rio/rioioctl.h>

#define PERMS 0

char    *progName       = NULL,
        *tty_device     = "/dev/ttyr000"; /* default tty */

int     fd,

main(int argc, char *argv[])
{
   if ( argc == 2 ) {          /* From command line arguments : */
        progName=argv[0];      /* get - program name             */
        tty_device=argv[1];    /* get - tty to use                  */
    }
    else if ( argc == 1 ) {
        progName=argv[0];      /* get - program name, use default tty */
    }
    else {
        printf("Usage - ioctltst 'tty'\n");
        exit(1);
    }

    /* echo program name */
    printf( "%s\n", progName );

    /* see if we can open tty */
    if (( fd = open( tty_device, O_RDWR, PERMS )) == -1 ) {
        perror("unable to open tty");
        exit(1);
    }

    /* ENABLE CTS flow control */
    if ( ioctl( fd, TCRIOCTSFLOWEN ) == -1 ) {
        perror("TCRIOCTSFLOWEN ioctl failed");
        exit(1);
    }

    /* ENABLE RTS flow control */
    if ( ioctl( fd, TCRIORTSFLOWEN ) == -1 ) {
        perror("TCRIORTSFLOWEN ioctl failed");
        exit(1);
    }

    printf( "end %s\n", progName );
    close(fd);
    exit(0);
}
```

Appendix C – Modem control ioctls

This appendix gives details of how to control and monitor RIO port modem signals.

The ioctls are defined in `/usr/include/rio/rioioctl.h`, which you will need to `'#include'` in your application if you want to control and monitor modem signals.

There are three ioctls :

TCRIOMBIS	-	the argument is the address of an `'int'` which contains a mask of the modem lines to be `set`.
TCRIOMBIC	-	the argument is the address of an `'int'` which contains a mask of the modem lines to be `cleared`.
TCRIOSTATE	-	the argument is the address of an `'int'` in which the current state of the modem signals will be returned.

The header file also defines bit masks for each of the applicable modem signals :

TCRIOM_RTS
TCRIOM_DTR
TCRIOM_DSR
TCRIOM_RI
TCRIOM_CTS
TCRIOM_DCD

Note : The RIO driver assumes a DTE port, so DTR & RTS are the only signals that may specified with bit clear/set ioctls.

An example of how to use these ioctls is given on the following pages. The application takes the name of a RIO device node as an argument, eg: `modem_lines /dev/ttyr000`

```
/* modem_lines.c */

#include <stdio.h>
#include <errno.h>
#include <fcntl.h>
#include <rio/rioioctl.h>

int printModemState();

char  *tty_device,
      *progName;

int   fd,
      modemLinesWrite,
      modemLinesRead;

main(int argc, char *argv[])
{
    progName   = argv[0];      /* get program name */

    if ( argc == 2 ) {
        tty_device=argv[1];    /* get - tty to use */
    }
    else {
        printf("Usage - %s /dev/ttyrnnn\n", progName );
        exit(1);
    }

    /* see if we can open tty */
    if (( fd = open( tty_device, O_RDWR, 0 )) == -1 ) {
        perror("unable to open tty");
        exit(1);
    }

    /* Get Initial State of modem lines */
    printf( "Initial state of modem lines :\n" );
    printModemState();

    /* Test TCRIOMBIC ioctl - clear DTR */
    modemLinesWrite = TCRIOM_DTR;
    sleep(1);
    printf( "ioctl: TCRIOMBIC DTR\n" );
    if ( ioctl( fd, TCRIOMBIC, &modemLinesWrite ) == -1 ) {
        perror("TCRIOMBIC ioctl failed");
        exit(1);
    }
    printModemState();

    /* Test TCRIOMBIS ioctl - set DTR signal */
    modemLinesWrite = TCRIOM_DTR;
    sleep(1);
    printf( "ioctl: TCRIOMBIS DTR\n" );
    if ( ioctl( fd, TCRIOMBIS, &modemLinesWrite ) == -1 ) {
        perror("TCRIOMBIS ioctl failed");
        exit(1);
    }
    printModemState();
```

```
    /* Test TCRIOMBIC ioctl - clear DTR & RTS signals */
    modemLinesWrite = TCRIOM_RTS | TCRIOM_DTR;
    sleep(1);
    printf( "ioctl: TCRIOMBIC DTR & RTS\n" );
    if ( ioctl( fd, TCRIOMBIC, &modemLinesWrite ) == -1 )
    {
        perror("TCRIOMBIC ioctl failed");
        exit(1);
    }
    printModemState();

    /* Test TCRIOMBIS ioctl - set DTR & RTS signals */
    modemLinesWrite = TCRIOM_RTS | TCRIOM_DTR;
    sleep(1);
    printf( "ioctl: TCRIOMBIS DTR & RTS\n" );
    if ( ioctl( fd, TCRIOMBIS, &modemLinesWrite ) == -1 )
    {
        perror("TCRIOMBIS ioctl failed");
        exit(1);
    }
    printModemState();

    close(fd);
    exit(0);
}

int printModemState()
{
    if ( ioctl(fd, TCRIOSTATE, &modemLinesRead ) == -1) {
        printf("%s: TCRIOSTATE ioctl failed (%d)\n", progName, errno);
        exit(1);
    }

    printf( "  modem lines :  DTR DSR RTS CTS CD  RI\n" );
    printf( "    (0x%4.4X)      %u   %u   %u   %u   %u   %u\n\n",
        modemLinesRead,
        (modemLinesRead & TCRIOM_DTR) ? 1 : 0,
        (modemLinesRead & TCRIOM_DSR) ? 1 : 0,
        (modemLinesRead & TCRIOM_RTS) ? 1 : 0,
        (modemLinesRead & TCRIOM_CTS) ? 1 : 0,
        (modemLinesRead & TCRIOM_DCD) ? 1 : 0,
        (modemLinesRead & TCRIOM_RI) ? 1 : 0    );
}
```

Appendix D – New Configuration Utilities

This appendix gives detailed descriptions of all the new configuration utilities that are installed with the driver from release 1.2.0 onwards.

The configuration file - /etc/rio/rio.cf – contains all the information the utilities may require to carry out configuration operations on the driver.

A default rio.cf file is installed with driver, but does not contain any host card or RTA configuration data. The default port entry (DEFPORT) is defined. Please read the following pages detailing the various fields in the rio.cf file first.

The driver will be booted at boot time by the command rioboot –f, which forces the driver to boot even if no configuration file is present. When the system comes up you should run :

```
rioboot -w
```

to get the configuration details of your RIO installation, redirect the output to a file, (not rio.cf). Add PORT configuration information, if all your ports on an RTA are to be configured the same way, you only need to add one PORT entry for each RTA.

An example rio.cf file from a system with 2 host cards and an RTA on each using the default configuration for all ports is shown on the next page. Note that the RTA names have been changed from the defaults given by the driver when booted. Setting up your rio.cf file like this will get you started.

Use

```
rioboot –c file –s
```

to syntax check your config file. When you are happy with the content of your config file, copy it to /etc/rio/rio.cf and run

```
rioboot –u
```

to update the driver with your configuration details. Then run

```
riomkdev -f
```

to make all the tty device nodes, (you may need the –f flag to force any existing RIO tty device nodes to be unlinked if you have installed on a system previously hosting a RIO set up).

Your RIO system is now ready for use.

Example rio.cf file :

```
#
# Current RIO driver configuration.
#
# File Created : Fri Feb 26 08:31:01 1999
#
#
# Host entry format is:
# HOST:<unique id>:<Device Name>:<master/slave>:<boot/noboot>:
#
# Rta entry format is:
# RTA:<unique id>:<Device Name>:<First Port>:<boot/noboot>:
#
# Port entry format is:
# PORT:<offset>:<tty dev>:<xp dev>:<xcps>:<xpon>:<xpoff>:ixany,ixon,lock,store,d
rain:
#
# Default port format is:
# DEFPORT:<offset>:<dev prfx>:<xp dev prfx>:<tty/modem>:<xcps>:<xpon>:<xpoff>:ix
any,ixon,lock,store,drain:
#
#

DEFPORT:0:/dev/ttyr%03d::tty:100:XPON:XPOFF:ixany:


HOST:d100000f:HOST 1:boot:
    RTA:94000944:RTA 1:0:boot:
        PORT:0:::tty:::::

HOST:da00000e:HOST 2:boot:
    RTA:94001a0b:RTA 2:16:boot:
        PORT:0:::tty:::::
```

NAME

rio.cf - file contains RIO device driver configuration information

DESCRIPTION

/etc/rio/rio.cf is an administrative file that contains information used by the RIO utilities to configure the RIO device driver and device nodes.

The file contains lines that are either comments or begin with one of a pre-defined set of keywords. Each line contains fields separated by the '**:**' character. A line beginning with the '**#**' character is defined as a comment.

The **rio.cf** file contains lines of the following types:

HOST A line beginning with this mnemonic is used to define a host card entry. The line contains the following fields:

unique num This field contains the unique number identifying the host card in the system.

name A string defining the name associated with this host card.

failsafe This field contains the string **master** or **slave** and defines whether the host card should attempt to boot any RTA devices connected to it. If this field is left blank then the default value is **master**.

bootflag This field contains the string **boot** or **noboot**. If the user sets this field to **noboot** then the device driver will not attempt to download the host card with its binary file. If this field is left blank then the default value is **boot**.

RTA A line beginning with this identifier is used to define an RTA device. The line contains the following fields:

unique num This field contains the unique number identifying the RTA in the system.

name This field contains a string defining the name associated with this RTA.

sysport This field contains the system port number to be assigned to the first port on the RTA. A system port number is any number from 0 to 504 and must be divisible by 8 leaving no remainder i.e. 0, 8, 16, 24 are all valid system port numbers, 43 is an invalid system port number. System port numbers are allocated in groups of 8 and a 16 port RTA consumes 2 consecutive system port groups. If this field is left blank then no system ports will be allocated to this device.

bootflag This field contains the string **boot** or **noboot**. If set to **noboot** then the RTA is not booted by the device driver and no ports on this system are assigned to it. The default for this field if left blank is **boot**.

DEFPORT A line beginning with this identifier is used to define the default values for the system ports. The line contains the following fields:

offset This field contains the numerical offset added to each system port when defining the default name for a device in the **riomkdev** command. The default for this field if left blank is **1**.

device prefix This field contains a string using the same format as the ANSI C *printf* function and defines the default name that will be given to a device node created using the **riomkdev** command e.g. the string "**/dev/term/rio%03d**" will create a device node for system port 30 of

"*/dev/term/rio030*" and a device node for system port 501 of "*/dev/term/rio501*". If this field is left blank the default string will be "**/dev/term/r03%d**".

xprint prefix This field contains a string using the same format as the ANSI C *printf* function and defines the default name that will be assigned to a transparent print device node created using the **riomkdev** command. If this field is left blank the default string will be "**/dev/term/r03%dp**".

device type This field contains the string **tty** or **modem** and defines the type of default device that should be created. A device of type of **tty** is a normal device and ignores the state of any modem signals during an open call and while the port is open. A device of type **modem** asserts the **DTR** output signal on an open and does not return from the open until the **DCD** input signal has been asserted. A **SIGHUP** signal is generated and sent to the controlling tty process when the **DCD** signal is de-asserted if the port is open. If this field is left blank the default is **tty**.

xprint cps This field contains a decimal number indicating the number of transparent print characters per second that should be sent to the port. Valid values for this field are between 10 and 1000. The default for this field if left blank is **100**.

xpoff string This fields contains the transparent print "off" string and is used to inform the terminal connected to the port that the following data is for the terminal. If this field is left blank the default string is "**XPOFF**".

xpon string This fields contains the transparent print "on" string and is used to inform the terminal connected to the port that the following data is for the transparent print port. If this field is left blank the default string is "**XPON**".

flags This field contains a combination of any of the following flags. Each flag must be separated by a '**,**' and the field must be ended with a '**:**'.

ixany Always open the port with the **termio** flag, *ixany* set.

ixon Always open the port with the **termio** flag, *ixon* set.

lock Lock the port settings. When this flag is set the RIO driver disallows any further attempts to configure a port using **termio**. See the **riolock** command for a more detailed description.

store Store the port settings across closes. When this flag is set the RIO driver keeps a copy of the port configuration and resets the port to this on each subsequent open. See the **riostore** command for a more detailed description.

PORT A line beginning with this identifier is used to define a unique set of port attributes to an individual port. This line must be preceded by a line of type **RTA** and the port referred to is assumed to be one of the ports connected to this RTA. If a port is multiply defined then the last entry in the configuration file will take precedence.

offset This field contains the numerical offset indicating which port on the previously specified RTA that the following attributes refer to. It is an error to leave this field blank.

device prefix This field contains a string uniquely identifying this port device. It is used by the **riomkdev** command to create or delete the device. If this field is left blank then the default tty device is created.

xprint prefix This field contains a string uniquely identifying this transparent print port device. It is used by the **riomkdev** command to create or delete the device. If this field is left blank then the default transparent print device is created.

device type This field contains string **tty** or **modem** and defines the type of tty device that should be created. See the **DEFPORT** description for more details.

xprint cps This field contains a decimal number indicating the number of transparent print characters per second that should be sent to the port. Valid values for this field are between 10 and 1000. If this field is left blank the default value is assumed.

xpoff string This fields contains the transparent print off string and is used to inform the terminal connected to the port that the following data is for the terminal. If this field is left blank the default string is assumed.

xpon string This fields contains the transparent print on string and is used to inform the terminal connected to the port that the following data is for the transparent print port. If this field is left blank the default string is assumed.

flags This field contains a combination of any of the flags **ixany**, **ixoff**, **lock** & **store**. See the **DEFPORT** description for more details. If this field is left blank then the default is assumed.

EXAMPLES

If you have a system with 2 host cards and 2 RTA's attached to each host the configuration file might look something like this:

```
#
# Example config file
#
DEFPORT:0:/dev/term/r%d:/dev/term/r%dp:tty:100:::ixany:

HOST:c11000011:First Host:master:boot:
   RTA:e30000d0:Software:0:boot:
      PORT:0:/dev/term/printer1::::::
   RTA:e30000d1:Hardware:8:boot:
      PORT:0:/dev/term/printer2::::::
HOST:c11000012:Second Host:master:boot:
   RTA:e30000d2:Marketing:16:boot:
      PORT:0:/dev/term/printer3::::::
   RTA:9400002d:Sales:24:boot:              # A 16 port RTA
      PORT:9:/dev/term/printer4::::::
```

The default setting for this site would produce tty devices starting at /dev/term/r0 and ending at /dev/term/r39. There are no modem devices in the system. There are 3 printers connected to port 0 of each of the 8 port RTA's and one printer connected to port 9 of the 16 port RTA. The printers have their own unique device names. The names of each of the RTA's are "Software", "Hardware", "Marketing" and "Sales".

If you have 3 modems connected to an RTA the configuration file might contain:

```
   RTA:e30000d0:Software:0:boot:
      PORT:0:/dev/term/modem1::modem::::
      PORT:1:/dev/term/modem2::modem::::
      PORT:2:/dev/term/modem3::modem::::
```

FILES

/etc/rio/rio.cf	RIO configuration file

NOTES

When the **rioadopt** command is used write this file, any comment lines in an existing file will be lost. The existing file will be moved to [filespec]**.old**.

SEE ALSO

rioboot, **riomkdev**, **rioreboot**, **rioshow, riolock, riounlock, riostore.**

NAME

rioboot - Initialise, update or save the RIO driver configuration.

SYNOPSIS

rioboot [-c *file*] [-h] [-v] [-s]

rioboot [-c *file*] [-h] [-v] -u

rioboot [-c *file*] [-h] [-f] [-v] -w

DESCRIPTION

During system start-up the RIO driver scans the hardware for available RIO host cards. It performs a comprehensive RAM test on all host cards that it finds and builds an internal table of the ones that pass their RAM test. The driver then exits its start-up routine leaving the host cards in a reset state.

Before any serial or transparent print ports can be accessed by the system the RIO driver must be initialised using the **rioboot** command. This command reads in the RIO configuration file and passes this information to the driver. When used to initialise the driver it performs three distinct functions:

1. Parses the RIO configuration file and initialises the driver with the host card and rta information found in this file. This includes the device unique number and name etc.
2. Reads in the RIO download code files and requests the driver download each of the host cards and any attached RTA's.
3. Parses the RIO configuration file and sets up the RIO ports with either the default settings or any custom settings defined within this file.

Once the **rioboot** command has initialised the RIO driver the attached serial and transparent print ports become accessible as long as the RTA they are connected to is booted.

The command **rioboot -u** can be used to update the driver with any changes that have been made to the configuration file after the driver has been initialised.

The command **rioboot -w** can be used to output the current configuration of the driver in the format of the configuration file. The output is the amalgamation of the current topology as supplied by the Driver and the current configuration file **(/etc/rio/rio.cf**). The output is sent to the standard output file descriptor and can be redirected using Shell commands. The command can be used to generate a new configuration file after adding a new host card and/or RTA's. If the **-c** option is used in conjunction with this option the file specified is the current configuration file.

Normal operation is for the command to execute silently and any errors are passed back through the exit code. The **-v** option, however, can be used to execute the command in *verbose* mode and any anomalies or errors will be reported to the standard error file descriptor.

The **-s** option can be used to perform a syntax check on the configuration file. This option is useful if you have made changes to the file and want to check that they are valid before commiting them to the Driver. When this option is selected the **-v** option is forced on to give as much information as possible about the file syntax.

The **-c** option can be used to specify an alternate configuration file.

The **-f** option can be used to force the Driver to attempt to boot even if no configuration file exists.

The **-h** option prints a usage message.

DIAGNOSTICS

The exit codes for **rioboot** are the following:

0	successful completion of task
1	Failed to initialise driver
2	Failed to download device binaries
3	Failed to set-up ports
4	Failed to save configuration
5	Terminated by signal

EXAMPLES

You could use rioboot to replace a malfunctioning RTA, (normally done using rioadopt) :

- Power off the malfunctioning RTA.
- Add the new RTA and power it on.
- Once the RTA has booted type "**rioshow**" and make a note of the old and new RTA unique numbers.
- Edit the file **/etc/rio/rio.cf** and change the unique number entry corresponding to the old RTA to the new RTA's unique number.
- Execute **rioboot -u**

FILES

/etc/rio/rio.cf	RIO configuration file
/etc/rio/rta.bin	Host card download code
/etc/rio/host.bin	Eight port RTA download code
/etc/init.d/rio (not LynxOS)	Invokes rioboot during system initialisation
/bin/rc (LynxOS only)	Invokes rioboot during system initialisation

NOTES

Creation of a configuration file will not generate any custom port settings.

When the **rioboot -w** option is executed any blank fields in custom port definitions will be expanded.

SEE ALSO

rioreboot, **rioshow**, **riomkdev**

NAME

rioadopt - Replace a malfunctioning RTA and update configuration automatically.

SYNOPSIS

rioadopt [-c *file*] [-h] [-v] -r

rioadopt [-c *file*] [-h] [-v] -i

DESCRIPTION

Updates configuration file and driver when a new RTA replaces one that is no longer connected. System operation continues without requiring a reboot.

May be invoked in “interactive” mode if the user requires the option of committing or rejecting the “adoption”.

May be invoked in “report only” mode if required.

The current content of the configuration file is compared with the configuration reported by the driver. If the driver reports that only one RTA is missing and only one new one is present then, unless invoked in “report only” mode, the new RTA will be adopted automatically if it is connected to the same host card as the missing one. If invoked in interactive mode, missing and new RTA unique Ids are displayed and the user is required to confirm or otherwise that the adoption should be committed to the driver.

The following are illegal combinations for RTA adoption and will force “report only” mode :

- New RTA(s) detected when none reported missing - add new RTA(s) to rio.cf and use rioboot -u.
- Missing RTA(s) reported and no new one to adopt.
- Missing RTA(s) reported and more than one new RTA detected - edit rio.cf and use rioboot -u.

If more than one RTA is reported missing and one new RTA is detected then, unless “interactive” mode is specified, “report only” mode is forced. This enables rioadopt to be invoked at boot time without the possibility of the boot process halting because rioadopt is waiting for user input. A report will displayed and the system administrator should login and run rioadopt interactively to select the missing RTA which the new one should take the place of.

When an RTA adoption takes place the configuration file is automatically updated, the existing configuration file is moved to [filespec].old. Future system reboots will automatically configure the adopted RTA.

Once an RTA has been adopted it assumes the name of the one it replaced. Any ports configured on the old RTA retain their device names and node numbers and continue to function exactly as before the adoption took place.

The **-c** option can be used to specify an alternate configuration file - **for read and write**.

The **-r** option selects “report only” mode.

The **-i** option causes rioadopt to operate interactively.

The **-v** Verbose mode. Error messages and warnings are sent to the standard error file descriptor.

The **-h** option prints a usage message.

DIAGNOSTICS

The exit codes for **rioadopt** are the following:

0	successful completion of task
1	Failed to initialise driver
4	Failed to save configuration

EXAMPLES

To replace a malfunctioning RTA :

- Power off the malfunctioning RTA.
- Add the new RTA and power it on.
- Once the RTA has booted type **rioadopt**.

FILES

/etc/rio/rio.cf	RIO configuration file

NOTES

When the **rioadopt** command is used to a new write the file, any comment lines in an existing file will be lost. Comments may be retrieved from the existing file which will be moved to [filespec]**.old**.

SEE ALSO

rioboot, rioreboot, **rioshow**

NAME

rioreboot - Reboot an RTA

SYNOPSIS

rioreboot [-h] [-v] *name|unique num*

rioreboot [-v] -n

DESCRIPTION

The **rioreboot** command requests the RIO driver to issue a reboot command packet to the specified RTA. The RTA can be specified by either its *name* or its *unique number*.

The **rioshow** command can be used to display the current machine configuration and the device names and unique numbers.

The normal operation of this command is silent but the **-v** option can be used to give a more verbose operation. All messages are sent to the standard error file descriptor.

The command **rioreboot -n** can be used to reboot any *network interconnected* RTA's present on the network. Network interconnected RTA's are ones that have been booted by an alternate host card to the one they are currently connected. This command is normally run after swapping out a host card. When used in conjunction with the **-v** option the command will display information about its current progress. The command attempts to reboot all the currently interconnected RTA's and will then wait until they have re-appeared on the network. It will then search for any more interconnected RTA's and issue reboot commands to any found. This process is repeated until there are no more interconnected RTA's or until a number of re-tries is exceeded.

The **-h** option prints a usage message.

EXAMPLES

If you have an RTA named "Ports 0-7" and want to reboot it then type:

rioreboot "Ports 0-7"

If you have a number of RTA's you want to reboot and you know either their unique numbers or their names then type:

rioreboot e3000ccf e00037da "Ports 0-7" 9400001b

DIAGNOSTICS

The exit codes for **rioreboot** are the following:

0 successful completion of task

1 Failed to reboot RTA.

NOTES

Data loss may be experienced when rebooting an active RTA.

If a network interconnected RTA is connected to more than one host card issuing a reboot will result in the RTA being rebooted but still network interconnected.

SEE ALSO

rioboot, **rioshow**, **riomkdev**

NAME

rioshow - Display current driver device configuration information.

SYNOPSIS

rioshow [-s] [-h] [-v]

DESCRIPTION

The **rioshow** command displays information about the current RIO driver configuration. The output displays the devices attached to the RIO driver one line at a time giving information about their unique number, driver id, system ports and name.

The **-v** option causes the command to print error messages to the standard error file descriptor.

The **-s** option inhibits the display of the column headings.

The **-h** option prints a usage message.

DIAGNOSTICS

The exit codes for **rioshow** are the following:

0 successful completion of task

1 Failed to retrieve information from RIO device driver.

EXAMPLES

A typical output from the rioshow command executed without any arguments should be:

Unique Num	**ID**	**Ports**	**Name**	
c1000011		**0**	**-1 - -1**	**Sbus Host card**
940000ab		**1**	**0 - 7**	**Ports 0 - 15**
e3000ccf		**3**	**16 - 23**	**Ports 16-23**

NOTES

The device driver does not have to have been initialised using the **rioboot** command before attempting to get the host card information.

SEE ALSO

rioboot, **rioreboot**, **riomkdev**

NAME

riomkdev - Create or delete RIO tty and transparent print devices.

SYNOPSIS

riomkdev [-c *file*] [-f] [-h] [-r*name*|*unique num*] [-v]

riomkdev [-c *file*] [-f] [-h] [-r*name*|*unique num*] [-v] -x

riomkdev [-c *file*] [-f] [-h] [-r*name*|*unique num*] [-v] -d

DESCRIPTION

The command **riomkdev** should be used to create and delete RIO specific device nodes.

The command parses the RIO configuration file **/etc/rio/rio.cf** and looks for lines beginning with **RTA:**, **DEFPORT:** or **PORT:** (see **rio.cf**). These lines are then used to define the device node name and the devices are created or deleted depending on the command line options.

The device major number is determined using the RTA sysport field and the associated host card control device major number. The device minor number is determined by the RTA sysport field.

The device node name is determined either by a **PORT:** entry for the corresponding device or it is deduced from the device's sysport number and the port name string in the **DEFPORT:** line.

Options

-c*file*	Specify an alternative configuration file.
-d	Delete devices. All specified devices shall be deleted if they exist.
-f	Force operation. Any existing device nodes shall be unlinked before the new devices are created.
-h	Print usage message and exit.
-r*name*|*unique num*	Specify the specific RTA whose devices should be created/deleted.
-v	Verbose mode. Error messages and warnings shall be sent to the standard error file descriptor.
-x	Create/delete transparent print devices.

DIAGNOSTICS

The exit codes for **riomkdev** are the following:

0 successful completion of task
1 Failed to create/delete devices

FILES

/etc/rio/rio.cf	RIO configuration file

NOTES

The command makes no attempt to determine if a particular device node is currently open or in use before deleting it.

SEE ALSO

rioboot, **rioreboot**, **rioshow**

NAME

riodelete - Remove an RTA entry from the RIO driver tables.

SYNOPSIS

riodelete [-f] [-h] [-v] *name*|*unique num*

DESCRIPTION

The command **riodelete** requests the RIO driver to remove the specified RTA device from its internal tables. The RTA can be specified by either its *name* or its *unique number*.

The **rioshow** command can be used to display the current RIO driver configuration including the devices names and unique numbers.

If an RTA is currently attached and booted then the command will not delete it unless the user has specified the **-f** option. This option will force the deletion of the device by removing it from the network and then requesting the RIO driver to delete its entry. The RTA is removed from the network by placing the RTA software in an infinite loop.

The normal operation of this command is silent but the **-v** option can be used to give a more verbose operation. All messages are sent to the standard error file descriptor.

The **-h** option prints a usage message to the standard error file descriptor.

DIAGNOSTICS

The exit codes for **riodelete** are the following:

0 successful completion of task
1 Command was unsuccessful

EXAMPLES

To remove an RTA named “Ports 0-7” type:

riodelete “Ports 0-7”

To force the removal of the RTA with unique number e3000045, type:

riodelete -f e3000045

NOTES

This command removes the RTA device from the driver tables but does not delete its entry from the RIO configuration file **/etc/rio/rio.cf**.

SEE ALSO

rioreboot, **rioshow**, **riomkdev, rioidentify**

NAME

rioidentify - Display current driver version information.

SYNOPSIS

rioidentify [-h] [-v] name|unique num

DESCRIPTION

The **rioidentify** command causes an RTA, specified either by its name or unique number, to flash its link LEDs amber. The operation continues until such time as the user terminates the program by pressing <return> in response to an appropriate prompt.

The **-v** option causes the command to print error messages to the standard error file descriptor.

The **-h** option prints a usage message.

DIAGNOSTICS

The exit codes for **rioidentify** are the following:

0 successful completion of task

1 Failed to retrieve information from RIO device driver.

NOTES

This command does not affect normal operation.

NAME

rioversion - Display current driver version information.

SYNOPSIS

rioversion [-h] [-v]

DESCRIPTION

The **rioversion** command displays the current version of the RIO device.

The **-v** option causes the command to print error messages to the standard error file descriptor.

The **-h** option prints a usage message.

DIAGNOSTICS

The exit codes for **rioversion** are the following:

0 successful completion of task

1 Failed to retrieve information from RIO device driver.

NOTES

The device driver does not have to have been initialised using the **rioboot** command before attempting to get the version information.

NAME

riostats - Display statistics from the RIO device driver.

SYNOPSIS

riostats [-h] [-v] [-e] ttyname

riostats [-h] [-v] [-d] ttyname

DESCRIPTION

Retrieves and displays device driver statistics on a per port basis.

The driver does not update its internal data until statistics gathering is “enabled”.

The driver may be inhibited from updating statistics information by “disabling” statistics gathering.

By default the output has column header information. This may be switched off,. For example, the riostats command may be invoked in a loop from a script which displays statistics information for all ports, (where the header information is only required to be output for the first port – the other port’s statistics being displayed directly underneath).

The **-e** enables driver statistics gathering..

The **-d** disables driver statistics gathering..

The **-v** Verbose mode. Error messages and warning are sent to the standard error file descriptor.

The **-h** option prints a usage message.

DIAGNOSTICS

The exit codes for **riostats** are the following:

0 successful completion of task

1 ‘stats’ ioctl failed.

EXAMPLES

To enable and start statistics gathering for the device /dev/ttyr000 :

- Type, riostats –e /dev/ttyr000.

To see statistics information for the device /dev/ttyr000 :

- Type, riostats /dev/ttyr000.

To disable statistics gathering for the device /dev/ttyr000 :

- Type, riostats –d /dev/ttyr000.

NOTES

Invoking riostats with the –d option does not reset the driver’s internal statistics data.

Appendix E – Dual-Host Fail-Safe

This appendix gives an example of how to set up Dual-Host Fail-Safe and subsequently swap RTAs from MASTER host to SLAVE host control. The scenario is the most basic Dual-Host set-up possible – two systems, each with one host card, both connected to one RTA.

1. Install RIO on both systems, leave the RTA switched off and reboot both systems as required following an installation.

2. As described in Appendix D, set up the RIO configuration file - /etc/rio/rio.cf – on both systems.

3. On the system which is to be the 'slave', set the `'bootflag'` field, for the host card only, to 'noboot'. Run rioboot –u again on the slave system.

4. Switch on the RTA, it should establish connections on the master system only. The slave system will report 'HOST 1 (A) connected to another network'.

5. To test Dual-Host Fail-Safe, remove the link between the RTA and the Master system's host card.

6. On the slave system, change the `'bootflag'` field back to 'boot' and run `rioboot -u`.

7. To get the slave system to take over the RTA, run `rioreboot -n,` (note that this command may take a while to complete, especially if there are many tiers of RTAs).