

RapidIO Subsystem Guide

Matt Porter

mporter@kernel.crashing.org
mporter@mvista.com

RapidIO Subsystem Guide

by Matt Porter

Copyright © 2005 MontaVista Software, Inc.

This documentation is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For more details see the file COPYING in the source distribution of Linux.

Table of Contents

1. Introduction.....	1
2. Known Bugs and Limitations	3
2.1. Bugs	3
2.2. Limitations	3
3. RapidIO driver interface.....	5
3.1. Functions.....	5
rio_local_read_config_32	5
rio_local_write_config_32	6
rio_local_read_config_16	7
rio_local_write_config_16	8
rio_local_read_config_8	9
rio_local_write_config_8	10
rio_read_config_32	11
rio_write_config_32	12
rio_read_config_16	13
rio_write_config_16	14
rio_read_config_8	15
rio_write_config_8	16
rio_send_doorbell	17
rio_init_mbox_res	18
rio_init_dbell_res	18
RIO_DEVICE	19
rio_add_outb_message	20
rio_add_inb_buffer	21
rio_get_inb_message	22
rio_name	23
rio_get_drvdata	24
rio_set_drvdata	25
rio_dev_get	26
rio_dev_put	27
rio_register_driver	27
rio_unregister_driver	28
rio_local_get_device_id	29
rio_request_inb_mbox	30
rio_release_inb_mbox	31
rio_request_outb_mbox	32
rio_release_outb_mbox	33
rio_request_inb_dbell	34
rio_release_inb_dbell	35
rio_request_outb_dbell	36

rio_release_outb_dbell.....	37
rio_request_inb_pwrite	38
rio_release_inb_pwrite.....	39
rio_inb_pwrite_handler.....	40
rio_get_asm.....	41
rio_get_device.....	42
4. Internals	45
4.1. Structures	45
struct rio_switch.....	45
struct rio_dev	47
struct rio_msg.....	49
struct rio_dbell	50
struct rio_mport.....	51
struct rio_net	53
struct rio_ops.....	54
struct rio_driver.....	56
struct rio_device_id.....	58
struct rio_switch_ops	59
4.2. Enumeration and Discovery	60
rio_get_device_id.....	60
rio_set_device_id	61
rio_local_set_device_id	62
rio_clear_locks.....	63
rio_enum_host.....	63
rio_device_has_destid.....	64
rio_release_dev	65
rio_is_switch.....	66
rio_switch_init	67
rio_add_device	68
rio_enable_rx_tx_port.....	69
rio_setup_device	70
rio_sport_is_active	71
rio_lock_device.....	72
rio_unlock_device.....	73
rio_route_add_entry	74
rio_route_get_entry.....	75
rio_get_host_deviceid_lock	77
rio_enum_peer	77
rio_enum_complete.....	79
rio_disc_peer.....	79
rio_mport_is_active	81
rio_alloc_net	81
rio_update_route_tables	82

rio_init_em.....	83
rio_pw_enable.....	84
rio_enum_mport.....	85
rio_build_route_tables	86
rio_enum_timeout.....	86
rio_disc_mport	87
4.3. Driver functionality	88
rio_setup_inb_dbell.....	88
rio_mport_get_physefb.....	89
rio_get_comptag	90
rio_set_port_lockout	91
rio_chk_dev_route	92
rio_mport_chk_dev_access.....	93
rio_chk_dev_access	94
rio_get_input_status.....	95
rio_clr_err_stopped.....	96
rio_mport_get_efb.....	96
rio_mport_get_feature.....	97
rio_std_route_add_entry	99
rio_std_route_get_entry	100
rio_std_route_clr_table	101
RIO_LOP_READ	102
RIO_LOP_WRITE	103
RIO_OP_READ.....	104
RIO_OP_WRITE.....	105
4.4. Device model support	106
rio_match_device	106
rio_device_probe.....	107
rio_device_remove.....	107
rio_match_bus.....	108
rio_bus_init	109
4.5. Sysfs support.....	110
rio_create_sysfs_dev_files	110
rio_remove_sysfs_dev_files.....	111
4.6. PPC32 support	112
fsl_local_config_read.....	112
fsl_local_config_write.....	113
fsl_rio_config_read	114
fsl_rio_config_write.....	115
fsl_rio_setup.....	117
5. Credits.....	119

Chapter 1. Introduction

RapidIO is a high speed switched fabric interconnect with features aimed at the embedded market. RapidIO provides support for memory-mapped I/O as well as message-based transactions over the switched fabric network. RapidIO has a standardized discovery mechanism not unlike the PCI bus standard that allows simple detection of devices in a network.

This documentation is provided for developers intending to support RapidIO on new architectures, write new drivers, or to understand the subsystem internals.

Chapter 2. Known Bugs and Limitations

2.1. Bugs

None. ;)

2.2. Limitations

1. Access/management of RapidIO memory regions is not supported
2. Multiple host enumeration is not supported

Chapter 3. RapidIO driver interface

Drivers are provided a set of calls in order to interface with the subsystem to gather info on devices, request/map memory region resources, and manage mailboxes/doorbells.

3.1. Functions

rio_local_read_config_32

LINUX

Kernel Hackers Manual January 2013

Name

`rio_local_read_config_32` — Read 32 bits from local configuration space

Synopsis

```
int rio_local_read_config_32 (struct rio_mport * port, u32
offset, u32 * data);
```

Arguments

port

Master port

offset

Offset into local configuration space

data

Pointer to read data into

Description

Reads 32 bits of data from the specified offset within the local device's configuration space.

rio_local_write_config_32

LINUX

Kernel Hackers Manual January 2013

Name

`rio_local_write_config_32` — Write 32 bits to local configuration space

Synopsis

```
int rio_local_write_config_32 (struct rio_mport * port, u32
offset, u32 data);
```

Arguments

port

Master port

offset

Offset into local configuration space

data

Data to be written

Description

Writes 32 bits of data to the specified offset within the local device's configuration space.

rio_local_read_config_16

LINUX

Kernel Hackers Manual January 2013

Name

`rio_local_read_config_16` — Read 16 bits from local configuration space

Synopsis

```
int rio_local_read_config_16 (struct rio_mport * port, u32
offset, u16 * data);
```

Arguments

port

Master port

offset

Offset into local configuration space

data

Pointer to read data into

Description

Reads 16 bits of data from the specified offset within the local device's configuration space.

rio_local_write_config_16

LINUX

Kernel Hackers Manual January 2013

Name

`rio_local_write_config_16` — Write 16 bits to local configuration space

Synopsis

```
int rio_local_write_config_16 (struct rio_mport * port, u32
offset, u16 data);
```

Arguments

port

Master port

offset

Offset into local configuration space

data

Data to be written

Description

Writes 16 bits of data to the specified offset within the local device's configuration space.

rio_local_read_config_8

LINUX

Kernel Hackers Manual January 2013

Name

`rio_local_read_config_8` — Read 8 bits from local configuration space

Synopsis

```
int rio_local_read_config_8 (struct rio_mport * port, u32
offset, u8 * data);
```

Arguments

port

Master port

offset

Offset into local configuration space

data

Pointer to read data into

Description

Reads 8 bits of data from the specified offset within the local device's configuration space.

rio_local_write_config_8

LINUX

Kernel Hackers Manual January 2013

Name

`rio_local_write_config_8` — Write 8 bits to local configuration space

Synopsis

```
int rio_local_write_config_8 (struct rio_mport * port, u32
offset, u8 data);
```

Arguments

port

Master port

offset

Offset into local configuration space

data

Data to be written

Description

Writes 8 bits of data to the specified offset within the local device's configuration space.

rio_read_config_32

LINUX

Kernel Hackers Manual January 2013

Name

`rio_read_config_32` — Read 32 bits from configuration space

Synopsis

```
int rio_read_config_32 (struct rio_dev * rdev, u32 offset, u32  
* data);
```

Arguments

rdev

RIO device

offset

Offset into device configuration space

data

Pointer to read data into

Description

Reads 32 bits of data from the specified offset within the RIO device's configuration space.

rio_write_config_32

LINUX

Kernel Hackers Manual January 2013

Name

`rio_write_config_32` — Write 32 bits to configuration space

Synopsis

```
int rio_write_config_32 (struct rio_dev * rdev, u32 offset,
u32 data);
```

Arguments

rdev

RIO device

offset

Offset into device configuration space

data

Data to be written

Description

Writes 32 bits of data to the specified offset within the RIO device's configuration space.

rio_read_config_16

LINUX

Kernel Hackers Manual January 2013

Name

`rio_read_config_16` — Read 16 bits from configuration space

Synopsis

```
int rio_read_config_16 (struct rio_dev * rdev, u32 offset, u16  
* data);
```

Arguments

rdev

RIO device

offset

Offset into device configuration space

data

Pointer to read data into

Description

Reads 16 bits of data from the specified offset within the RIO device's configuration space.

rio_write_config_16

LINUX

Kernel Hackers Manual January 2013

Name

`rio_write_config_16` — Write 16 bits to configuration space

Synopsis

```
int rio_write_config_16 (struct rio_dev * rdev, u32 offset,
u16 data);
```

Arguments

rdev

RIO device

offset

Offset into device configuration space

data

Data to be written

Description

Writes 16 bits of data to the specified offset within the RIO device's configuration space.

rio_read_config_8

LINUX

Kernel Hackers Manual January 2013

Name

`rio_read_config_8` — Read 8 bits from configuration space

Synopsis

```
int rio_read_config_8 (struct rio_dev * rdev, u32 offset, u8 * data);
```

Arguments

rdev

RIO device

offset

Offset into device configuration space

data

Pointer to read data into

Description

Reads 8 bits of data from the specified offset within the RIO device's configuration space.

rio_write_config_8

LINUX

Kernel Hackers Manual January 2013

Name

`rio_write_config_8` — Write 8 bits to configuration space

Synopsis

```
int rio_write_config_8 (struct rio_dev * rdev, u32 offset, u8 data);
```

Arguments

rdev

RIO device

offset

Offset into device configuration space

data

Data to be written

Description

Writes 8 bits of data to the specified offset within the RIO device's configuration space.

rio_send_doorbell

LINUX

Kernel Hackers Manual January 2013

Name

`rio_send_doorbell` — Send a doorbell message to a device

Synopsis

```
int rio_send_doorbell (struct rio_dev * rdev, u16 data);
```

Arguments

rdev

RIO device

data

Doorbell message data

Description

Send a doorbell message to a RIO device. The doorbell message has a 16-bit info field provided by the *data* argument.

rio_init_mbox_res

LINUX

Kernel Hackers Manual January 2013

Name

`rio_init_mbox_res` — Initialize a RIO mailbox resource

Synopsis

```
void rio_init_mbox_res (struct resource * res, int start, int  
end);
```

Arguments

res

resource struct

start

start of mailbox range

end

end of mailbox range

Description

This function is used to initialize the fields of a resource for use as a mailbox resource. It initializes a range of mailboxes using the start and end arguments.

rio_init_dbell_res

LINUX

Kernel Hackers Manual January 2013

Name

`rio_init_dbell_res` — Initialize a RIO doorbell resource

Synopsis

```
void rio_init_dbell_res (struct resource * res, u16 start, u16  
end);
```

Arguments

res

resource struct

start

start of doorbell range

end

end of doorbell range

Description

This function is used to initialize the fields of a resource for use as a doorbell resource. It initializes a range of doorbell messages using the start and end arguments.

RIO_DEVICE

LINUX

Kernel Hackers Manual January 2013

Name

RIO_DEVICE — macro used to describe a specific RIO device

Synopsis

```
RIO_DEVICE ( dev, ven );
```

Arguments

dev

the 16 bit RIO device ID

ven

the 16 bit RIO vendor ID

Description

This macro is used to create a struct `rio_device_id` that matches a specific device. The assembly vendor and assembly device fields will be set to `RIO_ANY_ID`.

rio_add_outb_message

LINUX

Name

`rio_add_outb_message` — Add RIO message to an outbound mailbox queue

Synopsis

```
int rio_add_outb_message (struct rio_mport * mport, struct
rio_dev * rdev, int mbox, void * buffer, size_t len);
```

Arguments

mport

RIO master port containing the outbound queue

rdev

RIO device the message is be sent to

mbox

The outbound mailbox queue

buffer

Pointer to the message buffer

len

Length of the message buffer

Description

Adds a RIO message buffer to an outbound mailbox queue for transmission.
Returns 0 on success.

rio_add_inb_buffer

LINUX

Kernel Hackers Manual January 2013

Name

`rio_add_inb_buffer` — Add buffer to an inbound mailbox queue

Synopsis

```
int rio_add_inb_buffer (struct rio_mport * mport, int mbox,  
void * buffer);
```

Arguments

mport

Master port containing the inbound mailbox

mbox

The inbound mailbox number

buffer

Pointer to the message buffer

Description

Adds a buffer to an inbound mailbox queue for reception. Returns 0 on success.

rio_get_inb_message

LINUX

Kernel Hackers Manual January 2013

Name

`rio_get_inb_message` — Get A RIO message from an inbound mailbox queue

Synopsis

```
void * rio_get_inb_message (struct rio_mport * mport, int mbox);
```

Arguments

mport

Master port containing the inbound mailbox

mbox

The inbound mailbox number

Description

Get a RIO message from an inbound mailbox queue. Returns 0 on success.

rio_name

LINUX

Name

`rio_name` — Get the unique RIO device identifier

Synopsis

```
const char * rio_name (struct rio_dev * rdev);
```

Arguments

rdev

RIO device

Description

Get the unique RIO device identifier. Returns the device identifier string.

rio_get_drvdata

LINUX

Name

`rio_get_drvdata` — Get RIO driver specific data

Synopsis

```
void * rio_get_drvdata (struct rio_dev * rdev);
```

Arguments

rdev

RIO device

Description

Get RIO driver specific data. Returns a pointer to the driver specific data.

rio_set_drvdata

LINUX

Kernel Hackers Manual January 2013

Name

`rio_set_drvdata` — Set RIO driver specific data

Synopsis

```
void rio_set_drvdata (struct rio_dev * rdev, void * data);
```

Arguments

rdev

RIO device

data

Pointer to driver specific data

Description

Set RIO driver specific data. device struct driver data pointer is set to the *data* argument.

rio_dev_get

LINUX

Kernel Hackers Manual January 2013

Name

`rio_dev_get` — Increments the reference count of the RIO device structure

Synopsis

```
struct rio_dev * rio_dev_get (struct rio_dev * rdev);
```

Arguments

rdev

RIO device being referenced

Description

Each live reference to a device should be refcounted.

Drivers for RIO devices should normally record such references in their `probe` methods, when they bind to a device, and release them by calling `rio_dev_put`, in their `disconnect` methods.

rio_dev_put

LINUX

Kernel Hackers Manual January 2013

Name

`rio_dev_put` — Release a use of the RIO device structure

Synopsis

```
void rio_dev_put (struct rio_dev * rdev);
```

Arguments

rdev

RIO device being disconnected

Description

Must be called when a user of a device is finished with it. When the last user of the device calls this function, the memory of the device is freed.

rio_register_driver

LINUX

Kernel Hackers Manual January 2013

Name

`rio_register_driver` — register a new RIO driver

Synopsis

```
int rio_register_driver (struct rio_driver * rdrv);
```

Arguments

rdrv

the RIO driver structure to register

Description

Adds a struct `rio_driver` to the list of registered drivers. Returns a negative value on error, otherwise 0. If no error occurred, the driver remains registered even if no device was claimed during registration.

rio_unregister_driver

LINUX

Name

`rio_unregister_driver` — unregister a RIO driver

Synopsis

```
void rio_unregister_driver (struct rio_driver * rdrv);
```

Arguments

rdrv

the RIO driver structure to unregister

Description

Deletes the struct `rio_driver` from the list of registered RIO drivers, gives it a chance to clean up by calling its `remove` function for each device it was responsible for, and marks those devices as driverless.

rio_local_get_device_id

LINUX

Name

`rio_local_get_device_id` — Get the base/extended device id for a port

Synopsis

```
ul6 rio_local_get_device_id (struct rio_mport * port);
```

Arguments

port

RIO master port from which to get the deviceid

Description

Reads the base/extended device id from the local device implementing the master port. Returns the 8/16-bit device id.

rio_request_inb_mbox

LINUX

Kernel Hackers Manual January 2013

Name

`rio_request_inb_mbox` — request inbound mailbox service

Synopsis

```
int rio_request_inb_mbox (struct rio_mport * mport, void *  
dev_id, int mbox, int entries, void (*minb) (struct rio_mport  
* mport, void *dev_id, int mbox, int slot));
```

Arguments

mport

RIO master port from which to allocate the mailbox resource

dev_id

Device specific pointer to pass on event

mbox

Mailbox number to claim

entries

Number of entries in inbound mailbox queue

minb

Callback to execute when inbound message is received

Description

Requests ownership of an inbound mailbox resource and binds a callback function to the resource. Returns 0 on success.

rio_release_inb_mbox

LINUX

Kernel Hackers Manual January 2013

Name

`rio_release_inb_mbox` — release inbound mailbox message service

Synopsis

```
int rio_release_inb_mbox (struct rio_mport * mport, int mbox);
```

Arguments

mport

RIO master port from which to release the mailbox resource

mbox

Mailbox number to release

Description

Releases ownership of an inbound mailbox resource. Returns 0 if the request has been satisfied.

rio_request_outb_mbox

LINUX

Kernel Hackers Manual January 2013

Name

`rio_request_outb_mbox` — request outbound mailbox service

Synopsis

```
int rio_request_outb_mbox (struct rio_mport * mport, void *  
dev_id, int mbox, int entries, void (*moutb) (struct rio_mport  
* mport, void *dev_id, int mbox, int slot));
```

Arguments

mport

RIO master port from which to allocate the mailbox resource

dev_id

Device specific pointer to pass on event

mbox

Mailbox number to claim

entries

Number of entries in outbound mailbox queue

moutb

Callback to execute when outbound message is sent

Description

Requests ownership of an outbound mailbox resource and binds a callback function to the resource. Returns 0 on success.

rio_release_outb_mbox

LINUX

Kernel Hackers Manual January 2013

Name

rio_release_outb_mbox — release outbound mailbox message service

Synopsis

```
int rio_release_outb_mbox (struct rio_mport * mport, int
mbox);
```

Arguments

mport

RIO master port from which to release the mailbox resource

mbox

Mailbox number to release

Description

Releases ownership of an inbound mailbox resource. Returns 0 if the request has been satisfied.

rio_request_inb_dbell

LINUX

Kernel Hackers Manual January 2013

Name

`rio_request_inb_dbell` — request inbound doorbell message service

Synopsis

```
int rio_request_inb_dbell (struct rio_mport * mport, void *
dev_id, ul6 start, ul6 end, void (*dinb) (struct rio_mport *
```



```
mport, void *dev_id, u16 src, u16 dst, u16 info));
```

Arguments

mport

RIO master port from which to allocate the doorbell resource

dev_id

Device specific pointer to pass on event

start

Doorbell info range start

end

Doorbell info range end

dinb

Callback to execute when doorbell is received

Description

Requests ownership of an inbound doorbell resource and binds a callback function to the resource. Returns 0 if the request has been satisfied.

rio_release_inb_dbell

LINUX

Kernel Hackers Manual January 2013

Name

rio_release_inb_dbell — release inbound doorbell message service

Synopsis

```
int rio_release_inb_dbell (struct rio_mport * mport, u16
start, u16 end);
```

Arguments

mport

RIO master port from which to release the doorbell resource

start

Doorbell info range start

end

Doorbell info range end

Description

Releases ownership of an inbound doorbell resource and removes callback from the doorbell event list. Returns 0 if the request has been satisfied.

rio_request_outb_dbell

LINUX

Kernel Hackers Manual January 2013

Name

`rio_request_outb_dbell` — request outbound doorbell message range

Synopsis

```
struct resource * rio_request_outb_dbell (struct rio_dev *  
rdev, u16 start, u16 end);
```

Arguments

rdev

RIO device from which to allocate the doorbell resource

start

Doorbell message range start

end

Doorbell message range end

Description

Requests ownership of a doorbell message range. Returns a resource if the request has been satisfied or `NULL` on failure.

rio_release_outb_dbell

LINUX

Kernel Hackers Manual January 2013

Name

`rio_release_outb_dbell` — release outbound doorbell message range

Synopsis

```
int rio_release_outb_dbell (struct rio_dev * rdev, struct
resource * res);
```

Arguments

rdev

RIO device from which to release the doorbell resource

res

Doorbell resource to be freed

Description

Releases ownership of a doorbell message range. Returns 0 if the request has been satisfied.

rio_request_inb_pwrite

LINUX

Kernel Hackers Manual January 2013

Name

`rio_request_inb_pwrite` — request inbound port-write message service

Synopsis

```
int rio_request_inb_pwrite (struct rio_dev * rdev, int
(*pwcback) (struct rio_dev *rdev, union rio_pw_msg *msg, int
```

```
step));
```

Arguments

rdev

RIO device to which register inbound port-write callback routine

pwcback

Callback routine to execute when port-write is received

Description

Binds a port-write callback function to the RapidIO device. Returns 0 if the request has been satisfied.

rio_release_inb_pwrite

LINUX

Kernel Hackers Manual January 2013

Name

`rio_release_inb_pwrite` — release inbound port-write message service

Synopsis

```
int rio_release_inb_pwrite (struct rio_dev * rdev);
```

Arguments

rdev

RIO device which registered for inbound port-write callback

Description

Removes callback from the `rio_dev` structure. Returns 0 if the request has been satisfied.

rio_inb_pwrite_handler

LINUX

Kernel Hackers Manual January 2013

Name

`rio_inb_pwrite_handler` — process inbound port-write message

Synopsis

```
int rio_inb_pwrite_handler (union rio_pw_msg * pw_msg);
```

Arguments

pw_msg

pointer to inbound port-write message

Description

Processes an inbound port-write message. Returns 0 if the request has been satisfied.

rio_get_asm

LINUX

Kernel Hackers Manual January 2013

Name

`rio_get_asm` — Begin or continue searching for a RIO device by vid/did/asm_vid/asm_did

Synopsis

```
struct rio_dev * rio_get_asm (u16 vid, u16 did, u16 asm_vid,  
u16 asm_did, struct rio_dev * from);
```

Arguments

vid

RIO vid to match or RIO_ANY_ID to match all vids

did

RIO did to match or RIO_ANY_ID to match all dids

asm_vid

RIO asm_vid to match or RIO_ANY_ID to match all asm_vids

asm_did

RIO asm_did to match or RIO_ANY_ID to match all asm_dids

from

Previous RIO device found in search, or `NULL` for new search

Description

Iterates through the list of known RIO devices. If a RIO device is found with a matching *vid*, *did*, *asm_vid*, *asm_did*, the reference count to the device is incremented and a pointer to its device structure is returned. Otherwise, `NULL` is returned. A new search is initiated by passing `NULL` to the *from* argument. Otherwise, if *from* is not `NULL`, searches continue from next device on the global list. The reference count for *from* is always decremented if it is not `NULL`.

rio_get_device

LINUX

Kernel Hackers Manual January 2013

Name

`rio_get_device` — Begin or continue searching for a RIO device by vid/did

Synopsis

```
struct rio_dev * rio_get_device (u16 vid, u16 did, struct
rio_dev * from);
```

Arguments

vid

RIO vid to match or `RIO_ANY_ID` to match all vids

did

RIO did to match or `RIO_ANY_ID` to match all dids

from

Previous RIO device found in search, or `NULL` for new search

Description

Iterates through the list of known RIO devices. If a RIO device is found with a matching *vid* and *did*, the reference count to the device is incremented and a pointer to its device structure is returned. Otherwise, `NULL` is returned. A new search is initiated by passing `NULL` to the *from* argument. Otherwise, if *from* is not `NULL`, searches continue from next device on the global list. The reference count for *from* is always decremented if it is not `NULL`.

Chapter 4. Internals

This chapter contains the autogenerated documentation of the RapidIO subsystem.

4.1. Structures

struct rio_switch

LINUX

Kernel Hackers Manual January 2013

Name

struct rio_switch — RIO switch info

Synopsis

```
struct rio_switch {
    struct list_head node;
    u16 switchid;
    u8 * route_table;
    u32 port_ok;
    int (* add_entry) (struct rio_mport *mport, u16 destid, u8 hopcount, u16
    int (* get_entry) (struct rio_mport *mport, u16 destid, u8 hopcount, u16
    int (* clr_table) (struct rio_mport *mport, u16 destid, u8 hopcount, u16
    int (* set_domain) (struct rio_mport *mport, u16 destid, u8 hopcount, u8
    int (* get_domain) (struct rio_mport *mport, u16 destid, u8 hopcount, u8
    int (* em_init) (struct rio_dev *dev);
    int (* em_handle) (struct rio_dev *dev, u8 swport);
    int (* sw_sysfs) (struct rio_dev *dev, int create);
    struct rio_dev * nextdev[0];
};
```

Members

node

Node in global list of switches

switchid

Switch ID that is unique across a network

route_table

Copy of switch routing table

port_ok

Status of each port (one bit per port) - OK=1 or UNINIT=0

add_entry

Callback for switch-specific route add function

get_entry

Callback for switch-specific route get function

clr_table

Callback for switch-specific clear route table function

set_domain

Callback for switch-specific domain setting function

get_domain

Callback for switch-specific domain get function

em_init

Callback for switch-specific error management init function

em_handle

Callback for switch-specific error management handler function

sw_sysfs

Callback that initializes switch-specific sysfs attributes

nextdev[0]

Array of per-port pointers to the next attached device

struct rio_dev

LINUX

Kernel Hackers Manual January 2013

Name

struct rio_dev — RIO device info

Synopsis

```
struct rio_dev {
    struct list_head global_list;
    struct list_head net_list;
    struct rio_net * net;
    u16 did;
    u16 vid;
    u32 device_rev;
    u16 asm_did;
    u16 asm_vid;
    u16 asm_rev;
    u16 efptr;
    u32 pef;
    u32 swpinfo;
    u32 src_ops;
    u32 dst_ops;
    u32 comp_tag;
    u32 phys_efptr;
    u32 em_efptr;
    u64 dma_mask;
    struct rio_driver * driver;
    struct device dev;
    struct resource riores[RIO_MAX_DEV_RESOURCES];
    int (* pwcback) (struct rio_dev *rdev, union rio_pw_msg *msg, int step);
    u16 destid;
    u8 hopcount;
    struct rio_dev * prev;
    struct rio_switch rswitch[0];
};
```

Members

global_list

Node in list of all RIO devices

net_list

Node in list of RIO devices in a network

net

Network this device is a part of

did

Device ID

vid

Vendor ID

device_rev

Device revision

asm_did

Assembly device ID

asm_vid

Assembly vendor ID

asm_rev

Assembly revision

efptr

Extended feature pointer

pef

Processing element features

swpinfo

Switch port info

src_ops

Source operation capabilities

`dst_ops`

Destination operation capabilities

`comp_tag`

RIO component tag

`phys_efptr`

RIO device extended features pointer

`em_efptr`

RIO Error Management features pointer

`dma_mask`

Mask of bits of RIO address this device implements

`driver`

Driver claiming this device

`dev`

Device model device

`riores[RIO_MAX_DEV_RESOURCES]`

RIO resources this device owns

`pwcback`

port-write callback function for this device

`destid`

Network destination ID (or associated `destid` for switch)

`hopcount`

Hopcount to this device

`prev`

Previous RIO device connected to the current one

`rswitch[0]`

struct `rio_switch` (if valid for this device)

struct rio_msg

LINUX

Kernel Hackers Manual January 2013

Name

struct rio_msg — RIO message event

Synopsis

```
struct rio_msg {
    struct resource * res;
    void (* mcback) (struct rio_mport * mport, void *dev_id, int mbox, int s
};
```

Members

res

Mailbox resource

mcback

Message event callback

struct rio_dbell

LINUX

Kernel Hackers Manual January 2013

Name

struct rio_dbell — RIO doorbell event

Synopsis

```
struct rio_dbell {
    struct list_head node;
    struct resource * res;
    void (* dinb) (struct rio_mport *mport, void *dev_id, u16 src, u16 dst,
    void * dev_id;
};
```

Members

node

Node in list of doorbell events

res

Doorbell resource

dinb

Doorbell event callback

dev_id

Device specific pointer to pass on event

struct rio_mport

LINUX

Kernel Hackers Manual January 2013

Name

struct rio_mport — RIO master port info

Synopsis

```
struct rio_mport {
```

```
struct list_head dbells;
struct list_head node;
struct list_head nnode;
struct resource iores;
struct resource riores[RIO_MAX_MPORT_RESOURCES];
struct rio_msg inb_msg[RIO_MAX_MBOX];
struct rio_msg outb_msg[RIO_MAX_MBOX];
int host_deviceid;
struct rio_ops * ops;
unsigned char id;
unsigned char index;
unsigned int sys_size;
enum rio_phy_type phy_type;
u32 phys_efptr;
unsigned char name[40];
void * priv;
};
```

Members

dbells

List of doorbell events

node

Node in global list of master ports

nnode

Node in network list of master ports

iores

I/O mem resource that this master port interface owns

riores[RIO_MAX_MPORT_RESOURCES]

RIO resources that this master port interfaces owns

inb_msg[RIO_MAX_MBOX]

RIO inbound message event descriptors

outb_msg[RIO_MAX_MBOX]

RIO outbound message event descriptors

<code>host_deviceid</code>	Host device ID associated with this master port
<code>ops</code>	configuration space functions
<code>id</code>	Port ID, unique among all ports
<code>index</code>	Port index, unique among all port interfaces of the same type
<code>sys_size</code>	RapidIO common transport system size
<code>phy_type</code>	RapidIO phy type
<code>phys_efptr</code>	RIO port extended features pointer
<code>name[40]</code>	Port name string
<code>priv</code>	Master port private data

struct rio_net

LINUX

Kernel Hackers Manual January 2013

Name

`struct rio_net` — RIO network info

Synopsis

```
struct rio_net {  
    struct list_head node;  
    struct list_head devices;  
    struct list_head mports;  
    struct rio_mport * hport;  
    unsigned char id;  
};
```

Members

node

Node in global list of RIO networks

devices

List of devices in this network

mports

List of master ports accessing this network

hport

Default port for accessing this network

id

RIO network ID

struct rio_ops

LINUX

Kernel Hackers Manual January 2013

Name

struct rio_ops — Low-level RIO configuration space operations

Synopsis

```
struct rio_ops {
    int (* lcread) (struct rio_mport *mport, int index, u32 offset, int len,
    int (* lcwrite) (struct rio_mport *mport, int index, u32 offset, int len,
    int (* cread) (struct rio_mport *mport, int index, u16 destid, u8 hopcount,
    int (* cwrite) (struct rio_mport *mport, int index, u16 destid, u8 hopcount,
    int (* dsend) (struct rio_mport *mport, int index, u16 destid, u16 data,
    int (* pwenable) (struct rio_mport *mport, int enable);
    int (* open_outb_mbox) (struct rio_mport *mport, void *dev_id, int mbox,
    void (* close_outb_mbox) (struct rio_mport *mport, int mbox);
    int (* open_inb_mbox) (struct rio_mport *mport, void *dev_id, int mbox,
    void (* close_inb_mbox) (struct rio_mport *mport, int mbox);
    int (* add_outb_message) (struct rio_mport *mport, struct rio_dev *rdev,
    int (* add_inb_buffer) (struct rio_mport *mport, int mbox, void *buf);
    void *(* get_inb_message) (struct rio_mport *mport, int mbox);
};
```

Members

lcread

Callback to perform local (master port) read of config space.

lcwrite

Callback to perform local (master port) write of config space.

cread

Callback to perform network read of config space.

cwrite

Callback to perform network write of config space.

dsend

Callback to send a doorbell message.

pwenable

Callback to enable/disable port-write message handling.

open_outb_mbox

Callback to initialize outbound mailbox.

`close_outb_mbox`

Callback to shut down outbound mailbox.

`open_inb_mbox`

Callback to initialize inbound mailbox.

`close_inb_mbox`

Callback to shut down inbound mailbox.

`add_outb_message`

Callback to add a message to an outbound mailbox queue.

`add_inb_buffer`

Callback to add a buffer to an inbound mailbox queue.

`get_inb_message`

Callback to get a message from an inbound mailbox queue.

struct rio_driver

LINUX

Kernel Hackers Manual January 2013

Name

`struct rio_driver` — RIO driver info

Synopsis

```
struct rio_driver {
    struct list_head node;
    char * name;
    const struct rio_device_id * id_table;
    int (* probe) (struct rio_dev * dev, const struct rio_device_id * id);
    void (* remove) (struct rio_dev * dev);
    int (* suspend) (struct rio_dev * dev, u32 state);
    int (* resume) (struct rio_dev * dev);
}
```

```
int (* enable_wake) (struct rio_dev * dev, u32 state, int enable);
struct device_driver driver;
};
```

Members

node

Node in list of drivers

name

RIO driver name

id_table

RIO device ids to be associated with this driver

probe

RIO device inserted

remove

RIO device removed

suspend

RIO device suspended

resume

RIO device awakened

enable_wake

RIO device enable wake event

driver

LDM driver struct

Description

Provides info on a RIO device driver for insertion/removal and power management purposes.

struct rio_device_id

LINUX

Kernel Hackers Manual January 2013

Name

`struct rio_device_id` — RIO device identifier

Synopsis

```
struct rio_device_id {  
    u16 did;  
    u16 vid;  
    u16 asm_did;  
    u16 asm_vid;  
};
```

Members

`did`

RIO device ID

`vid`

RIO vendor ID

`asm_did`

RIO assembly device ID

`asm_vid`

RIO assembly vendor ID

Description

Identifies a RIO device based on both the device/vendor IDs and the assembly device/vendor IDs.

struct rio_switch_ops

LINUX

Kernel Hackers Manual January 2013

Name

`struct rio_switch_ops` — Per-switch operations

Synopsis

```
struct rio_switch_ops {
    u16 vid;
    u16 did;
    int (* init_hook) (struct rio_dev *rdev, int do_enum);
};
```

Members

`vid`

RIO vendor ID

`did`

RIO device ID

`init_hook`

Callback that performs switch device initialization

Description

Defines the operations that are necessary to initialize/control a particular RIO switch device.

4.2. Enumeration and Discovery

rio_get_device_id

LINUX

Kernel Hackers Manual January 2013

Name

`rio_get_device_id` — Get the base/extended device id for a device

Synopsis

```
u16 rio_get_device_id (struct rio_mport * port, u16 destid, u8  
hopcount);
```

Arguments

port

RIO master port

destid

Destination ID of device

hopcount

Hopcount to device

Description

Reads the base/extended device id from a device. Returns the 8/16-bit device ID.

rio_set_device_id

LINUX

Kernel Hackers Manual January 2013

Name

`rio_set_device_id` — Set the base/extended device id for a device

Synopsis

```
void rio_set_device_id (struct rio_mport * port, u16 destid,  
u8 hopcount, u16 did);
```

Arguments

port

RIO master port

destid

Destination ID of device

hopcount

Hopcount to device

did

Device ID value to be written

Description

Writes the base/extended device id from a device.

rio_local_set_device_id

LINUX

Kernel Hackers Manual January 2013

Name

`rio_local_set_device_id` — Set the base/extended device id for a port

Synopsis

```
void rio_local_set_device_id (struct rio_mport * port, u16  
did);
```

Arguments

port

RIO master port

did

Device ID value to be written

Description

Writes the base/extended device id from a device.

rio_clear_locks

LINUX

Kernel Hackers Manual January 2013

Name

`rio_clear_locks` — Release all host locks and signal enumeration complete

Synopsis

```
int rio_clear_locks (struct rio_mport * port);
```

Arguments

port

Master port to issue transaction

Description

Marks the component tag CSR on each device with the enumeration complete flag. When complete, it then release the host locks on each device. Returns 0 on success or `-EINVAL` on failure.

rio_enum_host

LINUX

Kernel Hackers Manual January 2013

Name

`rio_enum_host` — Set host lock and initialize host destination ID

Synopsis

```
int rio_enum_host (struct rio_mport * port);
```

Arguments

port

Master port to issue transaction

Description

Sets the local host master port lock and destination ID register with the host device ID value. The host device ID value is provided by the platform. Returns 0 on success or -1 on failure.

rio_device_has_destid

LINUX

Name

`rio_device_has_destid` — Test if a device contains a destination ID register

Synopsis

```
int rio_device_has_destid (struct rio_mport * port, int
src_ops, int dst_ops);
```

Arguments

port

Master port to issue transaction

src_ops

RIO device source operations

dst_ops

RIO device destination operations

Description

Checks the provided *src_ops* and *dst_ops* for the necessary transaction capabilities that indicate whether or not a device will implement a destination ID register. Returns 1 if true or 0 if false.

rio_release_dev

LINUX

Name

`rio_release_dev` — Frees a RIO device struct

Synopsis

```
void rio_release_dev (struct device * dev);
```

Arguments

dev

LDM device associated with a RIO device struct

Description

Gets the RIO device struct associated a RIO device struct. The RIO device struct is freed.

rio_is_switch

LINUX

Name

`rio_is_switch` — Tests if a RIO device has switch capabilities

Synopsis

```
int rio_is_switch (struct rio_dev * rdev);
```

Arguments

rdev

RIO device

Description

Gets the RIO device Processing Element Features register contents and tests for switch capabilities. Returns 1 if the device is a switch or 0 if it is not a switch. The RIO device struct is freed.

rio_switch_init

LINUX

Kernel Hackers Manual January 2013

Name

`rio_switch_init` — Sets switch operations for a particular vendor switch

Synopsis

```
void rio_switch_init (struct rio_dev * rdev, int do_enum);
```

Arguments

rdev

RIO device

do_enum

Enumeration/Discovery mode flag

Description

Searches the RIO switch ops table for known switch types. If the vid and did match a switch table entry, then call switch initialization routine to setup switch-specific routines.

rio_add_device

LINUX

Kernel Hackers Manual January 2013

Name

`rio_add_device` — Adds a RIO device to the device model

Synopsis

```
int rio_add_device (struct rio_dev * rdev);
```

Arguments

rdev

RIO device

Description

Adds the RIO device to the global device list and adds the RIO device to the RIO device list. Creates the generic sysfs nodes for an RIO device.

rio_enable_rx_tx_port

LINUX

Kernel Hackers Manual January 2013

Name

`rio_enable_rx_tx_port` — enable input receiver and output transmitter of given port

Synopsis

```
int rio_enable_rx_tx_port (struct rio_mport * port, int local,
u16 destid, u8 hopcount, u8 port_num);
```

Arguments

port

Master port associated with the RIO network

local

local=1 select local port otherwise a far device is reached

destid

Destination ID of the device to check host bit

hopcount

Number of hops to reach the target

port_num

Port (-number on switch) to enable on a far end device

Description

Returns 0 or 1 from on General Control Command and Status Register (EXT_PTR+0x3C)

rio_setup_device

LINUX

Kernel Hackers Manual January 2013

Name

`rio_setup_device` — Allocates and sets up a RIO device

Synopsis

```
struct rio_dev * rio_setup_device (struct rio_net * net,  
struct rio_mport * port, u16 destid, u8 hopcount, int  
do_enum);
```

Arguments

net

RIO network

port

Master port to send transactions

destid

Current destination ID

hopcount

Current hopcount

do_enum

Enumeration/Discovery mode flag

Description

Allocates a RIO device and configures fields based on configuration space contents. If device has a destination ID register, a destination ID is either assigned in enumeration mode or read from configuration space in discovery mode. If the device has switch capabilities, then a switch is allocated and configured appropriately. Returns a pointer to a RIO device on success or NULL on failure.

rio_sport_is_active

LINUX

Kernel Hackers Manual January 2013

Name

`rio_sport_is_active` — Tests if a switch port has an active connection.

Synopsis

```
int rio_sport_is_active (struct rio_mport * port, u16 destid,
u8 hopcount, int sport);
```

Arguments

port

Master port to send transaction

destid

Associated destination ID for switch

hopcount

Hopcount to reach switch

sport

Switch port number

Description

Reads the port error status CSR for a particular switch port to determine if the port has an active link. Returns `RIO_PORT_N_ERR_STS_PORT_OK` if the port is active or 0 if it is inactive.

rio_lock_device

LINUX

Kernel Hackers Manual January 2013

Name

`rio_lock_device` — Acquires host device lock for specified device

Synopsis

```
int rio_lock_device (struct rio_mport * port, u16 destid, u8  
hopcount, int wait_ms);
```

Arguments

port

Master port to send transaction

destid

Destination ID for device/switch

hopcount

Hopcount to reach switch

wait_ms

Max wait time in msec (0 = no timeout)

Description

Attempts to acquire host device lock for specified device Returns 0 if device lock acquired or EINVAL if timeout expires.

rio_unlock_device

LINUX

Kernel Hackers Manual January 2013

Name

`rio_unlock_device` — Releases host device lock for specified device

Synopsis

```
int rio_unlock_device (struct rio_mport * port, u16 destid, u8  
hopcount);
```

Arguments

port

Master port to send transaction

destid

Destination ID for device/switch

hopcount

Hopcount to reach switch

Description

Returns 0 if device lock released or EINVAL if fails.

rio_route_add_entry

LINUX

Kernel Hackers Manual January 2013

Name

`rio_route_add_entry` — Add a route entry to a switch routing table

Synopsis

```
int rio_route_add_entry (struct rio_dev * rdev, u16 table, u16
route_destid, u8 route_port, int lock);
```

Arguments

rdev

RIO device

table

Routing table ID

route_destid

Destination ID to be routed

route_port

Port number to be routed

lock

lock switch device flag

Description

Calls the switch specific `add_entry` method to add a route entry on a switch. The route table can be specified using the *table* argument if a switch has per port routing tables or the normal use is to specify all tables (or the global table) by passing `RIO_GLOBAL_TABLE` in *table*. Returns 0 on success or `-EINVAL` on failure.

rio_route_get_entry

LINUX

Name

`rio_route_get_entry` — Read a route entry in a switch routing table

Synopsis

```
int rio_route_get_entry (struct rio_dev * rdev, u16 table, u16
route_destid, u8 * route_port, int lock);
```

Arguments

rdev

RIO device

table

Routing table ID

route_destid

Destination ID to be routed

route_port

Pointer to read port number into

lock

lock switch device flag

Description

Calls the switch specific `get_entry` method to read a route entry in a switch. The route table can be specified using the *table* argument if a switch has per port routing tables or the normal use is to specific all tables (or the global table) by passing `RIO_GLOBAL_TABLE` in *table*. Returns 0 on success or `-EINVAL` on failure.

rio_get_host_deviceid_lock

LINUX

Kernel Hackers Manual January 2013

Name

`rio_get_host_deviceid_lock` — Reads the Host Device ID Lock CSR on a device

Synopsis

```
u16 rio_get_host_deviceid_lock (struct rio_mport * port, u8  
hopcount);
```

Arguments

port

Master port to send transaction

hopcount

Number of hops to the device

Description

Used during enumeration to read the Host Device ID Lock CSR on a RIO device. Returns the value of the lock register.

rio_enum_peer

LINUX

Kernel Hackers Manual January 2013

Name

`rio_enum_peer` — Recursively enumerate a RIO network through a master port

Synopsis

```
int rio_enum_peer (struct rio_net * net, struct rio_mport *  
port, u8 hopcount, struct rio_dev * prev, int prev_port);
```

Arguments

net

RIO network being enumerated

port

Master port to send transactions

hopcount

Number of hops into the network

prev

Previous RIO device connected to the enumerated one

prev_port

Port on previous RIO device

Description

Recursively enumerates a RIO network. Transactions are sent via the master port passed in *port*.

rio_enum_complete

LINUX

Kernel Hackers Manual January 2013

Name

`rio_enum_complete` — Tests if enumeration of a network is complete

Synopsis

```
int rio_enum_complete (struct rio_mport * port);
```

Arguments

port

Master port to send transaction

Description

Tests the PGCCSR discovered bit for non-zero value (enumeration complete flag). Return 1 if enumeration is complete or 0 if enumeration is incomplete.

rio_disc_peer

LINUX

Kernel Hackers Manual January 2013

Name

`rio_disc_peer` — Recursively discovers a RIO network through a master port

Synopsis

```
int rio_disc_peer (struct rio_net * net, struct rio_mport *  
port, u16 destid, u8 hopcount, struct rio_dev * prev, int  
prev_port);
```

Arguments

net

RIO network being discovered

port

Master port to send transactions

destid

Current destination ID in network

hopcount

Number of hops into the network

prev

previous `rio_dev`

prev_port

previous port number

Description

Recursively discovers a RIO network. Transactions are sent via the master port passed in *port*.

rio_mport_is_active

LINUX

Kernel Hackers Manual January 2013

Name

`rio_mport_is_active` — Tests if master port link is active

Synopsis

```
int rio_mport_is_active (struct rio_mport * port);
```

Arguments

port

Master port to test

Description

Reads the port error status CSR for the master port to determine if the port has an active link. Returns `RIO_PORT_N_ERR_STS_PORT_OK` if the master port is active or 0 if it is inactive.

rio_alloc_net

LINUX

Kernel Hackers Manual January 2013

Name

`rio_alloc_net` — Allocate and configure a new RIO network

Synopsis

```
struct rio_net * rio_alloc_net (struct rio_mport * port);
```

Arguments

port

Master port associated with the RIO network

Description

Allocates a RIO network structure, initializes per-network list heads, and adds the associated master port to the network list of associated master ports. Returns a RIO network pointer on success or `NULL` on failure.

rio_update_route_tables

LINUX

Name

`rio_update_route_tables` — Updates route tables in switches

Synopsis

```
void rio_update_route_tables (struct rio_mport * port);
```

Arguments

port

Master port associated with the RIO network

Description

For each enumerated device, ensure that each switch in a system has correct routing entries. Add routes for devices that were unknown during the first enumeration pass through the switch.

rio_init_em

LINUX

Name

`rio_init_em` — Initializes RIO Error Management (for switches)

Synopsis

```
void rio_init_em (struct rio_dev * rdev);
```

Arguments

rdev

RIO device

Description

For each enumerated switch, call device-specific error management initialization routine (if supplied by the switch driver).

rio_pw_enable

LINUX

Kernel Hackers Manual January 2013

Name

`rio_pw_enable` — Enables/disables port-write handling by a master port

Synopsis

```
void rio_pw_enable (struct rio_mport * port, int enable);
```

Arguments

port

Master port associated with port-write handling

enable

1=enable, 0=disable

rio_enum_mport

LINUX

Kernel Hackers Manual January 2013

Name

`rio_enum_mport` — Start enumeration through a master port

Synopsis

```
int rio_enum_mport (struct rio_mport * mport);
```

Arguments

mport

Master port to send transactions

Description

Starts the enumeration process. If somebody has enumerated our master port device, then give up. If not and we have an active link, then start recursive peer enumeration. Returns 0 if enumeration succeeds or `-EBUSY` if enumeration fails.

rio_build_route_tables

LINUX

Kernel Hackers Manual January 2013

Name

`rio_build_route_tables` — Generate route tables from switch route entries

Synopsis

```
void rio_build_route_tables ( void );
```

Arguments

void

no arguments

Description

For each switch device, generate a route table by copying existing route entries from the switch.

rio_enum_timeout

LINUX

Name

`rio_enum_timeout` — Signal that enumeration timed out

Synopsis

```
void rio_enum_timeout (unsigned long data);
```

Arguments

data

Address of timeout flag.

Description

When the enumeration complete timer expires, set a flag that signals to the discovery process that enumeration did not complete in a sane amount of time.

rio_disc_mport

LINUX

Name

`rio_disc_mport` — Start discovery through a master port

Synopsis

```
int rio_disc_mport (struct rio_mport * mport);
```

Arguments

mport

Master port to send transactions

Description

Starts the discovery process. If we have an active link, then wait for the signal that enumeration is complete. When enumeration completion is signaled, start recursive peer discovery. Returns 0 if discovery succeeds or `-EBUSY` on failure.

4.3. Driver functionality

`rio_setup_inb_dbell`

LINUX

Kernel Hackers Manual January 2013

Name

`rio_setup_inb_dbell` — bind inbound doorbell callback

Synopsis

```
int rio_setup_inb_dbell (struct rio_mport * mport, void *
dev_id, struct resource * res, void (*dinb) (struct rio_mport
* mport, void *dev_id, u16 src, u16 dst, u16 info));
```

Arguments

mport

RIO master port to bind the doorbell callback

dev_id

Device specific pointer to pass on event

res

Doorbell message resource

dinb

Callback to execute when doorbell is received

Description

Adds a doorbell resource/callback pair into a port's doorbell event list. Returns 0 if the request has been satisfied.

rio_mport_get_physefb

LINUX

Name

`rio_mport_get_physefb` — Helper function that returns register offset for Physical Layer Extended Features Block.

Synopsis

```
u32 rio_mport_get_physefb (struct rio_mport * port, int local,  
u16 destid, u8 hopcount);
```

Arguments

port

Master port to issue transaction

local

Indicate a local master port or remote device access

destid

Destination ID of the device

hopcount

Number of switch hops to the device

`rio_get_comptag`

LINUX

Name

`rio_get_comptag` — Begin or continue searching for a RIO device by component tag

Synopsis

```
struct rio_dev * rio_get_comptag (u32 comp_tag, struct rio_dev  
* from);
```

Arguments

comp_tag

RIO component tag to match

from

Previous RIO device found in search, or `NULL` for new search

Description

Iterates through the list of known RIO devices. If a RIO device is found with a matching *comp_tag*, a pointer to its device structure is returned. Otherwise, `NULL` is returned. A new search is initiated by passing `NULL` to the *from* argument. Otherwise, if *from* is not `NULL`, searches continue from next device on the global list.

`rio_set_port_lockout`

LINUX

Name

`rio_set_port_lockout` — Sets/clears LOCKOUT bit (RIO EM 1.3) for a switch port.

Synopsis

```
int rio_set_port_lockout (struct rio_dev * rdev, u32 pnum, int lock);
```

Arguments

rdev

Pointer to RIO device control structure

pnum

Switch port number to set LOCKOUT bit

lock

Operation : set (=1) or clear (=0)

rio_chk_dev_route

LINUX

Name

`rio_chk_dev_route` — Validate route to the specified device.

Synopsis

```
int rio_chk_dev_route (struct rio_dev * rdev, struct rio_dev
** nrdev, int * npnum);
```

Arguments

rdev

RIO device failed to respond

nrdev

Last active device on the route to rdev

npnum

nrdev's port number on the route to rdev

Description

Follows a route to the specified RIO device to determine the last available device (and corresponding RIO port) on the route.

rio_mport_chk_dev_access

LINUX

Kernel Hackers Manual January 2013

Name

`rio_mport_chk_dev_access` — Validate access to the specified device.

Synopsis

```
int rio_mport_chk_dev_access (struct rio_mport * mport, u16  
destid, u8 hopcount);
```

Arguments

mport

Master port to send transactions

destid

Device destination ID in network

hopcount

Number of hops into the network

rio_chk_dev_access

LINUX

Kernel Hackers Manual January 2013

Name

`rio_chk_dev_access` — Validate access to the specified device.

Synopsis

```
int rio_chk_dev_access (struct rio_dev * rdev);
```

Arguments

rdev

Pointer to RIO device control structure

rio_get_input_status

LINUX

Kernel Hackers Manual January 2013

Name

`rio_get_input_status` — Sends a Link-Request/Input-Status control symbol and returns link-response (if requested).

Synopsis

```
int rio_get_input_status (struct rio_dev * rdev, int pnum, u32
* lnkresp);
```

Arguments

rdev

RIO devive to issue Input-status command

pnum

Device port number to issue the command

lnkresp

Response from a link partner

rio_clr_err_stopped

LINUX

Kernel Hackers Manual January 2013

Name

`rio_clr_err_stopped` — Clears port Error-stopped states.

Synopsis

```
int rio_clr_err_stopped (struct rio_dev * rdev, u32 pnum, u32  
err_status);
```

Arguments

rdev

Pointer to RIO device control structure

pnum

Switch port number to clear errors

err_status

port error status (if 0 reads register from device)

rio_mport_get_efb

LINUX

Name

`rio_mport_get_efb` — get pointer to next extended features block

Synopsis

```
u32 rio_mport_get_efb (struct rio_mport * port, int local, u16  
destid, u8 hopcount, u32 from);
```

Arguments

port

Master port to issue transaction

local

Indicate a local master port or remote device access

destid

Destination ID of the device

hopcount

Number of switch hops to the device

from

Offset of current Extended Feature block header (if 0 starts from
ExtFeaturePtr)

`rio_mport_get_feature`

LINUX

Name

`rio_mport_get_feature` — query for devices' extended features

Synopsis

```
u32 rio_mport_get_feature (struct rio_mport * port, int local,  
u16 destid, u8 hopcount, int ftr);
```

Arguments

port

Master port to issue transaction

local

Indicate a local master port or remote device access

destid

Destination ID of the device

hopcount

Number of switch hops to the device

ftr

Extended feature code

Description

Tell if a device supports a given RapidIO capability. Returns the offset of the requested extended feature block within the device's RIO configuration space or 0 in case the device does not support it. Possible values for *ftr*:

`RIO_EFB_PAR_EP_ID` LP/LVDS EP Devices

`RIO_EFB_PAR_EP_REC_ID` LP/LVDS EP Recovery Devices

RIO_EFB_PAR_EP_FREE_ID LP/LVDS EP Free Devices

RIO_EFB_SER_EP_ID LP/Serial EP Devices

RIO_EFB_SER_EP_REC_ID LP/Serial EP Recovery Devices

RIO_EFB_SER_EP_FREE_ID LP/Serial EP Free Devices

rio_std_route_add_entry

LINUX

Kernel Hackers Manual January 2013

Name

`rio_std_route_add_entry` — Add switch route table entry using standard registers defined in RIO specification rev.1.3

Synopsis

```
int rio_std_route_add_entry (struct rio_mport * mport, u16
destid, u8 hopcount, u16 table, u16 route_destid, u8
route_port);
```

Arguments

mport

Master port to issue transaction

destid

Destination ID of the device

hopcount

Number of switch hops to the device

table

routing table ID (global or port-specific)

route_destid

destID entry in the RT

route_port

destination port for specified destID

rio_std_route_get_entry

LINUX

Kernel Hackers Manual January 2013

Name

`rio_std_route_get_entry` — Read switch route table entry (port number) associated with specified destID using standard registers defined in RIO specification rev.1.3

Synopsis

```
int rio_std_route_get_entry (struct rio_mport * mport, u16
destid, u8 hopcount, u16 table, u16 route_destid, u8 *
route_port);
```

Arguments

mport

Master port to issue transaction

destid

Destination ID of the device

hopcount

Number of switch hops to the device

table

routing table ID (global or port-specific)

route_destid

destID entry in the RT

route_port

returned destination port for specified destID

rio_std_route_clr_table

LINUX

Kernel Hackers Manual January 2013

Name

`rio_std_route_clr_table` — Clear swotch route table using standard registers defined in RIO specification rev.1.3.

Synopsis

```
int rio_std_route_clr_table (struct rio_mport * mport, u16
destid, u8 hopcount, u16 table);
```

Arguments

mport

Master port to issue transaction

destid

Destination ID of the device

hopcount

Number of switch hops to the device

table

routing table ID (global or port-specific)

RIO_LOP_READ

LINUX

Kernel Hackers Manual January 2013

Name

RIO_LOP_READ — Generate rio_local_read_config_* functions

Synopsis

```
RIO_LOP_READ ( size, type, len );
```

Arguments

size

Size of configuration space read (8, 16, 32 bits)

type

C type of value argument

len

Length of configuration space read (1, 2, 4 bytes)

Description

Generates `rio_local_read_config_*` functions used to access configuration space registers on the local device.

RIO_LOP_WRITE

LINUX

Kernel Hackers Manual January 2013

Name

`RIO_LOP_WRITE` — Generate `rio_local_write_config_*` functions

Synopsis

```
RIO_LOP_WRITE ( size, type, len );
```

Arguments

size

Size of configuration space write (8, 16, 32 bits)

type

C type of value argument

len

Length of configuration space write (1, 2, 4 bytes)

Description

Generates `rio_local_write_config_*` functions used to access configuration space registers on the local device.

RIO_OP_READ

LINUX

Kernel Hackers Manual January 2013

Name

RIO_OP_READ — Generate `rio_mport_read_config_*` functions

Synopsis

```
RIO_OP_READ ( size, type, len );
```

Arguments

size

Size of configuration space read (8, 16, 32 bits)

type

C type of value argument

len

Length of configuration space read (1, 2, 4 bytes)

Description

Generates `rio_mport_read_config_*` functions used to access configuration space registers on the local device.

RIO_OP_WRITE

LINUX

Kernel Hackers Manual January 2013

Name

`RIO_OP_WRITE` — Generate `rio_mport_write_config_*` functions

Synopsis

```
RIO_OP_WRITE ( size, type, len );
```

Arguments

size

Size of configuration space write (8, 16, 32 bits)

type

C type of value argument

len

Length of configuration space write (1, 2, 4 bytes)

Description

Generates `rio_mport_write_config_*` functions used to access configuration space registers on the local device.

4.4. Device model support

rio_match_device

LINUX

Kernel Hackers Manual January 2013

Name

`rio_match_device` — Tell if a RIO device has a matching RIO device id structure

Synopsis

```
const struct rio_device_id * rio_match_device (const struct
rio_device_id * id, const struct rio_dev * rdev);
```

Arguments

id

the RIO device id structure to match against

rdev

the RIO device structure to match against

Description

Used from driver probe and bus matching to check whether a RIO device matches a device id structure provided by a RIO driver. Returns the matching struct `rio_device_id` or `NULL` if there is no match.

rio_device_probe

LINUX

Kernel Hackers Manual January 2013

Name

`rio_device_probe` — Tell if a RIO device structure has a matching RIO device id structure

Synopsis

```
int rio_device_probe (struct device * dev);
```

Arguments

dev

the RIO device structure to match against

Description

return 0 and set `rio_dev->driver` when drv claims `rio_dev`, else error

rio_device_remove

LINUX

Kernel Hackers Manual January 2013

Name

`rio_device_remove` — Remove a RIO device from the system

Synopsis

```
int rio_device_remove (struct device * dev);
```

Arguments

dev

the RIO device structure to match against

Description

Remove a RIO device from the system. If it has an associated driver, then run the `driver_remove` method. Then update the reference count.

rio_match_bus

LINUX

Name

`rio_match_bus` — Tell if a RIO device structure has a matching RIO driver device id structure

Synopsis

```
int rio_match_bus (struct device * dev, struct device_driver *  
drv);
```

Arguments

dev

the standard device structure to match against

drv

the standard driver structure containing the ids to match against

Description

Used by a driver to check whether a RIO device present in the system is in its list of supported devices. Returns 1 if there is a matching struct `rio_device_id` or 0 if there is no match.

`rio_bus_init`

LINUX

Name

`rio_bus_init` — Register the RapidIO bus with the device model

Synopsis

```
int rio_bus_init ( void );
```

Arguments

void

no arguments

Description

Registers the RIO bus device and RIO bus type with the Linux device model.

4.5. Sysfs support

`rio_create_sysfs_dev_files`

LINUX

Name

`rio_create_sysfs_dev_files` — create RIO specific sysfs files

Synopsis

```
int rio_create_sysfs_dev_files (struct rio_dev * rdev);
```

Arguments

rdev

device whose entries should be created

Description

Create files when *rdev* is added to sysfs.

rio_remove_sysfs_dev_files

LINUX

Kernel Hackers Manual January 2013

Name

`rio_remove_sysfs_dev_files` — cleanup RIO specific sysfs files

Synopsis

```
void rio_remove_sysfs_dev_files (struct rio_dev * rdev);
```

Arguments

rdev

device whose entries we should free

Description

Cleanup when *rdev* is removed from sysfs.

4.6. PPC32 support

fsl_local_config_read

LINUX

Kernel Hackers Manual January 2013

Name

`fsl_local_config_read` — Generate a MPC85xx local config space read

Synopsis

```
int fsl_local_config_read (struct rio_mport * mport, int
index, u32 offset, int len, u32 * data);
```

Arguments

mport

RapidIO master port info

index

ID of RapdiIO interface

offset

Offset into configuration space

len

Length (in bytes) of the maintenance transaction

data

Value to be read into

Description

Generates a MPC85xx local configuration space read. Returns 0 on success or `-EINVAL` on failure.

fsl_local_config_write

LINUX

Kernel Hackers Manual January 2013

Name

`fsl_local_config_write` — Generate a MPC85xx local config space write

Synopsis

```
int fsl_local_config_write (struct rio_mport * mport, int
index, u32 offset, int len, u32 data);
```

Arguments

mport

RapidIO master port info

index

ID of RapidIO interface

offset

Offset into configuration space

len

Length (in bytes) of the maintenance transaction

data

Value to be written

Description

Generates a MPC85xx local configuration space write. Returns 0 on success or `-EINVAL` on failure.

fsl_rio_config_read

LINUX

Kernel Hackers Manual January 2013

Name

`fsl_rio_config_read` — Generate a MPC85xx read maintenance transaction

Synopsis

```
int fsl_rio_config_read (struct rio_mport * mport, int index,  
u16 destid, u8 hopcount, u32 offset, int len, u32 * val);
```

Arguments

mport

RapidIO master port info

index

ID of RapidIO interface

destid

Destination ID of transaction

hopcount

Number of hops to target device

offset

Offset into configuration space

len

Length (in bytes) of the maintenance transaction

val

Location to be read into

Description

Generates a MPC85xx read maintenance transaction. Returns 0 on success or `-EINVAL` on failure.

fsl_rio_config_write

LINUX

Kernel Hackers Manual January 2013

Name

`fsl_rio_config_write` — Generate a MPC85xx write maintenance transaction

Synopsis

```
int fsl_rio_config_write (struct rio_mport * mport, int index,  
u16 destid, u8 hopcount, u32 offset, int len, u32 val);
```

Arguments

mport

RapidIO master port info

index

ID of RapdiIO interface

destid

Destination ID of transaction

hopcount

Number of hops to target device

offset

Offset into configuration space

len

Length (in bytes) of the maintenance transaction

val

Value to be written

Description

Generates an MPC85xx write maintenance transaction. Returns 0 on success or `-EINVAL` on failure.

fsl_rio_setup

LINUX

Kernel Hackers Manual January 2013

Name

`fsl_rio_setup` — Setup Freescale PowerPC RapidIO interface

Synopsis

```
int fsl_rio_setup (struct platform_device * dev);
```

Arguments

dev

platform_device pointer

Description

Initializes MPC85xx RapidIO hardware interface, configures master port with system-specific info, and registers the master port with the RapidIO subsystem.

Chapter 5. Credits

The following people have contributed to the RapidIO subsystem directly or indirectly:

1. Matt Porter<mporter@kernel.crashing.org>
2. Randy Vinson<rvinson@mvista.com>
3. Dan Malek<dan@embeddedalley.com>

The following people have contributed to this document:

1. Matt Porter<mporter@kernel.crashing.org>

