RapidIO Subsystem Guide

Matt Porter

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

Table of Contents

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

Table of Contents

Bugs

None. ;)

Limitations

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

Chapter 3. RapidIO driver interface

Table of Contents

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.

Functions

Name

rio_local_read_config_32 — Read 32 bits from local configuration space

Synopsis

```
int rio_local_read_config_32 (port,
                              offset,
                              data);

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.

Name

rio_local_write_config_32 — Write 32 bits to local configuration space

Synopsis

```
int rio_local_write_config_32 (port,
                               offset,
                               data);

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.

Name

rio_local_read_config_16 — Read 16 bits from local configuration space

Synopsis

```
int rio_local_read_config_16 (port,
                              offset,
                              data);

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.

Name

rio_local_write_config_16 — Write 16 bits to local configuration space

Synopsis

```
int rio_local_write_config_16 (port,
                               offset,
                               data);

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.

Name

rio_local_read_config_8 — Read 8 bits from local configuration space

Synopsis

```
int rio_local_read_config_8 (port,
                             offset,
                             data);

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.

Name

rio_local_write_config_8 — Write 8 bits to local configuration space

Synopsis

```
int rio_local_write_config_8 (port,
                              offset,
                              data);
```

```
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.

Name

rio_read_config_32 — Read 32 bits from configuration space

Synopsis

```
int rio_read_config_32 (rdev,
                        offset,
                        data);
```

```
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.

Name

rio_write_config_32 — Write 32 bits to configuration space

Synopsis

```
int rio_write_config_32 (rdev,
                         offset,
                         data);
```

```
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.

Name

rio_read_config_16 — Read 16 bits from configuration space

Synopsis

```
int rio_read_config_16 (rdev,
                        offset,
                        data);

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.

Name

rio_write_config_16 — Write 16 bits to configuration space

Synopsis

```
int rio_write_config_16 (rdev,
                         offset,
                         data);
```

```
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.

Name

rio_read_config_8 — Read 8 bits from configuration space

Synopsis

```
int rio_read_config_8 (rdev,
                       offset,
                       data);

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.

Name

rio_write_config_8 — Write 8 bits to configuration space

Synopsis

```
int rio_write_config_8 (rdev,
                        offset,
                        data);
```

```
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.

Name

rio_send_doorbell — Send a doorbell message to a device

Synopsis

```
int rio_send_doorbell (rdev,
                       data);

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.

Name

rio_init_mbox_res — Initialize a RIO mailbox resource

Synopsis

```
void rio_init_mbox_res (res,
                        start,
                        end);

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.

Name

rio_init_dbell_res — Initialize a RIO doorbell resource

Synopsis

```
void rio_init_dbell_res (res,
                         start,
                         end);

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.

Name

RIO_DEVICE — macro used to describe a specific RIO device

Synopsis

```
RIO_DEVICE (dev,
            ven);

 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`.

Name

rio_add_outb_message — Add RIO message to an outbound mailbox queue

Synopsis

```
int rio_add_outb_message (mport,
                          rdev,
                          mbox,
                          buffer,
                          len);

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.

Name

rio_add_inb_buffer — Add buffer to an inbound mailbox queue

Synopsis

```
int rio_add_inb_buffer (mport,
                        mbox,
                        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.

Name

rio_get_inb_message — Get A RIO message from an inbound mailbox queue

Synopsis

```
void * rio_get_inb_message (mport,
                            mbox);

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.

Name

rio_name — Get the unique RIO device identifier

Synopsis

```
const char * rio_name (rdev);

struct rio_dev *  rdev;
```

Arguments

rdev

RIO device

Description

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

Name

rio_get_drvdata — Get RIO driver specific data

Synopsis

```
void * rio_get_drvdata (rdev);

struct rio_dev *  rdev;
```

Arguments

rdev

RIO device

Description

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

Name

rio_set_drvdata — Set RIO driver specific data

Synopsis

```
void rio_set_drvdata (rdev,
                      data);

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.

Name

rio_dev_get — Increments the reference count of the RIO device structure

Synopsis

```
struct rio_dev * rio_dev_get (rdev);

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.

Name

rio_dev_put — Release a use of the RIO device structure

Synopsis

```
void rio_dev_put (rdev);

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.

Name

rio_register_driver — register a new RIO driver

Synopsis

```
int rio_register_driver (rdrv);

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.

Name

rio_unregister_driver — unregister a RIO driver

Synopsis

```
void rio_unregister_driver (rdrv);

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.

Name

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

Synopsis

```
u16 rio_local_get_device_id (port);

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.

Name

rio_request_inb_mbox — request inbound mailbox service

Synopsis

```
int rio_request_inb_mbox (mport,
                          dev_id,
                          mbox,
                          entries,
                          minb);

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.

Name

rio_release_inb_mbox — release inbound mailbox message service

Synopsis

```
int rio_release_inb_mbox (mport,
                          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.

Name

rio_request_outb_mbox — request outbound mailbox service

Synopsis

```
int rio_request_outb_mbox (mport,
                           dev_id,
                           mbox,
                           entries,
                           moutb);
```

```
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.

Name

rio_release_outb_mbox — release outbound mailbox message service

Synopsis

```
int rio_release_outb_mbox (mport,
                           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.

Name

rio_request_inb_dbell — request inbound doorbell message service

Synopsis

```
int rio_request_inb_dbell (mport,
                           dev_id,
                           start,
                           end,
                           dinb);

struct rio_mport *  mport;
void *              dev_id;
u16                 start;
u16                 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.

Name

rio_release_inb_dbell — release inbound doorbell message service

Synopsis

```
int rio_release_inb_dbell (mport,
                           start,
                           end);

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.

Name

rio_request_outb_dbell — request outbound doorbell message range

Synopsis

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

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.

Name

rio_release_outb_dbell — release outbound doorbell message range

Synopsis

```
int rio_release_outb_dbell (rdev,
                            res);

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.

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 (vid,
                              did,
                              asm_vid,
                              asm_did,
                              from);

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 incrememted 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`.

Name

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

Synopsis

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

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 incrememted 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

Table of Contents

This chapter contains the autogenerated documentation of the RapidIO subsystem.

Structures

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;
  u64 dma_mask;
  struct rio_switch * rswitch;
  struct rio_driver * driver;
  struct device dev;
  struct resource riores[RIO_MAX_DEV_RESOURCES];
  u16 destid;
};
```

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

dma_mask

Mask of bits of RIO address this device implements

rswitch

Pointer to struct rio_switch if valid for this device

driver

Driver claiming this device

dev

Device model device

riores[RIO_MAX_DEV_RESOURCES]

RIO resources this device owns

destid

Network destination ID

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 slot);
};
```

Members

res

Mailbox resource

mcback

Message event callback

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, u16 info);
  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

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;
  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

host_deviceid

Host device ID associated with this master port

ops

configuration space functions

id

Port ID, unique among all ports

index

Port index, unique among all port interfaces of the same type

sys_size

RapidIO common transport system size

phy_type

RapidIO phy type

name[40]

Port name string

priv

Master port private data

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

Name

struct rio_switch — RIO switch info

Synopsis

```
struct rio_switch {
  struct list_head node;
  u16 switchid;
  u16 hopcount;
  u16 destid;
  u8 * route_table;
  int (* add_entry) (struct rio_mport * mport, u16 destid, u8 hopcount,u16 table, u16 route_destid, u8 route_port);
  int (* get_entry) (struct rio_mport * mport, u16 destid, u8 hopcount,u16 table, u16 route_destid, u8 * route_port);
};
```

Members

node

Node in global list of switches

switchid

Switch ID that is unique across a network

hopcount

Hopcount to this switch

destid

Associated destid in the path

route_table

Copy of switch routing table

add_entry

Callback for switch-specific route add function

get_entry

Callback for switch-specific route get function

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,u32 *data);
  int (* lcwrite) (struct rio_mport *mport, int index, u32 offset, int len,u32 data);
  int (* cread) (struct rio_mport *mport, int index, u16 destid,u8 hopcount, u32 offset, int len, u32 *data);
  int (* cwrite) (struct rio_mport *mport, int index, u16 destid,u8 hopcount, u32 offset, int len, u32 data);
  int (* dsend) (struct rio_mport *mport, int index, u16 destid, u16 data);
};
```

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.

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.

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.

Name

struct rio_route_ops — Per-switch route operations

Synopsis

```
struct rio_route_ops {
  u16 vid;
  u16 did;
  int (* add_hook) (struct rio_mport * mport, u16 destid, u8 hopcount,u16 table, u16 route_destid, u8 route_port);
  int (* get_hook) (struct rio_mport * mport, u16 destid, u8 hopcount,u16 table, u16 route_destid, u8 * route_port);
};
```

Members

vid

RIO vendor ID

did

RIO device ID

add_hook

Callback that adds a route entry

get_hook

Callback that gets a route entry

Description

Defines the operations that are necessary to manipulate the route tables for a particular RIO switch device.

Enumeration and Discovery

Name

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

Synopsis

u16 **rio_get_device_id** (*port*,
 destid,

```
                         hopcount);

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.

Name

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

Synopsis

```
void rio_set_device_id (port,
                        destid,
                        hopcount,
                        did);

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.

Name

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

Synopsis

```
void rio_local_set_device_id (port,
                              did);

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.

Name

rio_clear_locks — Release all host locks and signal enumeration complete

Synopsis

```
int rio_clear_locks (port);

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.

Name

rio_enum_host — Set host lock and initialize host destination ID

Synopsis

```
int rio_enum_host (port);

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.

Name

rio_device_has_destid — Test if a device contains a destination ID register

Synopsis

```
int rio_device_has_destid (port,
                           src_ops,
                           dst_ops);

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.

Name

rio_release_dev — Frees a RIO device struct

Synopsis

```
void rio_release_dev (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.

Name

rio_is_switch — Tests if a RIO device has switch capabilities

Synopsis

```
int rio_is_switch (rdev);

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.

Name

rio_route_set_ops — Sets routing operations for a particular vendor switch

Synopsis

```
void rio_route_set_ops (rdev);

struct rio_dev *  rdev;
```

Arguments

rdev

RIO device

Description

Searches the RIO route ops table for known switch types. If the vid and did match a switch table entry, then set the `add_entry` and `get_entry` ops to the table entry values.

Name

rio_add_device — Adds a RIO device to the device model

Synopsis

```
int rio_add_device (rdev);

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.

Name

rio_setup_device — Allocates and sets up a RIO device

Synopsis

```
struct rio_dev * rio_setup_device (net,
                                   port,
                                   destid,
                                   hopcount,
                                   do_enum);

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.

Name

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

Synopsis

```
int rio_sport_is_active (port,
                         destid,
                         hopcount,
                         sport);

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 `PORT_N_ERR_STS_PORT_OK` if the port is active or `0` if it is inactive.

Name

rio_route_add_entry — Add a route entry to a switch routing table

Synopsis

```
int rio_route_add_entry (mport,
                         rswitch,
                         table,
                         route_destid,
                         route_port);

struct rio_mport *   mport;
struct rio_switch *  rswitch;
u16                  table;
u16                  route_destid;
u8                   route_port;
```

Arguments

mport

Master port to send transaction

rswitch

Switch device

table

Routing table ID

route_destid

Destination ID to be routed

route_port

Port number to be routed

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 specific all tables (or the global table) by passing `RIO_GLOBAL_TABLE` in *table*. Returns `0` on success or `-EINVAL` on failure.

Name

rio_route_get_entry — Read a route entry in a switch routing table

Synopsis

```
int rio_route_get_entry (mport,
                         rswitch,
                         table,
                         route_destid,
                         route_port);

struct rio_mport *   mport;
struct rio_switch *  rswitch;
u16                  table;
u16                  route_destid;
u8 *                 route_port;
```

Arguments

mport

Master port to send transaction

rswitch

Switch device

table

Routing table ID

route_destid

Destination ID to be routed

route_port

Pointer to read port number into

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.

Name

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

Synopsis

```
u16 rio_get_host_deviceid_lock (port,
                                hopcount);

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.

Name

rio_get_swpinfo_inport — Gets the ingress port number

Synopsis

```
u8 rio_get_swpinfo_inport (mport,
                           destid,
                           hopcount);

struct rio_mport *  mport;
u16                 destid;
u8                  hopcount;
```

Arguments

mport

Master port to send transaction

destid

Destination ID associated with the switch

hopcount

Number of hops to the device

Description

Returns port number being used to access the switch device.

Name

rio_get_swpinfo_tports — Gets total number of ports on the switch

Synopsis

```
u8 rio_get_swpinfo_tports (mport,
                           destid,
                           hopcount);

struct rio_mport *  mport;
u16                 destid;
u8                  hopcount;
```

Arguments

mport

Master port to send transaction

destid

Destination ID associated with the switch

hopcount

Number of hops to the device

Description

Returns total numbers of ports implemented by the switch device.

Name

rio_net_add_mport — Add a master port to a RIO network

Synopsis

```
void rio_net_add_mport (net,
                        port);

struct rio_net *    net;
struct rio_mport *  port;
```

Arguments

net

RIO network

port

Master port to add

Description

Adds a master port to the network list of associated master ports..

Name

rio_enum_peer — Recursively enumerate a RIO network through a master port

Synopsis

```
int rio_enum_peer (net,
                   port,
                   hopcount);

struct rio_net *    net;
struct rio_mport *  port;
u8                  hopcount;
```

Arguments

net

RIO network being enumerated

port

Master port to send transactions

hopcount

Number of hops into the network

Description

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

Name

rio_enum_complete — Tests if enumeration of a network is complete

Synopsis

```
int rio_enum_complete (port);

struct rio_mport *  port;
```

Arguments

`port`

  Master port to send transaction

Description

Tests the Component Tag CSR for presence of the magic enumeration complete flag. Return `1` if enumeration is complete or `0` if enumeration is incomplete.

Name

rio_disc_peer — Recursively discovers a RIO network through a master port

Synopsis

```
int rio_disc_peer (net,
                   port,
                   destid,
                   hopcount);

struct rio_net *    net;
struct rio_mport *  port;
u16                 destid;
u8                  hopcount;
```

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

Description

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

Name

rio_mport_is_active — Tests if master port link is active

Synopsis

```
int rio_mport_is_active (port);

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 `PORT_N_ERR_STS_PORT_OK` if the master port is active or `0` if it is inactive.

Name

rio_alloc_net — Allocate and configure a new RIO network

Synopsis

```
struct rio_net * rio_alloc_net (port);

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.

Name

rio_update_route_tables — Updates route tables in switches

Synopsis

```
void rio_update_route_tables (port);

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 where unknown dirung the first enumeration pass through the switch.

Name

rio_enum_mport — Start enumeration through a master port

Synopsis

```
int rio_enum_mport (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.

Name

rio_build_route_tables — Generate route tables from switch route entries

Synopsis

```
void rio_build_route_tables (void);

 void;
```

Arguments

void

no arguments

Description

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

Name

rio_enum_timeout — Signal that enumeration timed out

Synopsis

```
void rio_enum_timeout (data);

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.

Name

rio_disc_mport — Start discovery through a master port

Synopsis

```
int rio_disc_mport (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.

Driver functionality

Name

rio_setup_inb_dbell — bind inbound doorbell callback

Synopsis

```
int rio_setup_inb_dbell (mport,
                         dev_id,
                         res,
                         dinb);

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.

Name

rio_mport_get_feature — query for devices' extended features

Synopsis

```
u32 rio_mport_get_feature (port,
                           local,
                           destid,
                           hopcount,
                           ftr);

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

Name

RIO_LOP_READ — Generate rio_local_read_config_* functions

Synopsis

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

 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.

Name

RIO_LOP_WRITE — Generate rio_local_write_config_* functions

Synopsis

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

 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.

Name

RIO_OP_READ — Generate rio_mport_read_config_* functions

Synopsis

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

 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.

Name

RIO_OP_WRITE — Generate rio_mport_write_config_* functions

Synopsis

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

 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.

Device model support

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 (id,
                                               rdev);

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.

Name

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

Synopsis

```
int rio_device_probe (dev);

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

---

Name

rio_device_remove — Remove a RIO device from the system

Synopsis

```
int rio_device_remove (dev);

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.

Name

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

Synopsis

```
int rio_match_bus (dev,
                   drv);

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.

Name

rio_bus_init — Register the RapidIO bus with the device model

Synopsis

```
int rio_bus_init (void);

 void;
```

Arguments

void

no arguments

Description

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

Sysfs support

Name

rio_create_sysfs_dev_files — create RIO specific sysfs files

Synopsis

```
int rio_create_sysfs_dev_files (rdev);

struct rio_dev *  rdev;
```

Arguments

rdev

device whose entries should be created

Description

Create files when *rdev* is added to sysfs.

Name

rio_remove_sysfs_dev_files — cleanup RIO specific sysfs files

Synopsis

```
void rio_remove_sysfs_dev_files (rdev);

struct rio_dev *  rdev;
```

Arguments

rdev

device whose entries we should free

Description

Cleanup when *rdev* is removed from sysfs.

PPC32 support

Name

rio_hw_add_outb_message — Add message to the MPC85xx outbound message queue

Synopsis

```
int rio_hw_add_outb_message (mport,
                             rdev,
                             mbox,
                             buffer,
                             len);

struct rio_mport *  mport;
struct rio_dev *    rdev;
int                 mbox;
void *              buffer;
```

```
size_t              len;
```

Arguments

mport

Master port with outbound message queue

rdev

Target of outbound message

mbox

Outbound mailbox

buffer

Message to add to outbound queue

len

Length of message

Description

Adds the *buffer* message to the MPC85xx outbound message queue. Returns `0` on success or `-EINVAL` on failure.

Name

rio_hw_add_inb_buffer — Add buffer to the MPC85xx inbound message queue

Synopsis

```
int rio_hw_add_inb_buffer (mport,
                           mbox,
                           buf);

struct rio_mport *  mport;
int                 mbox;
void *              buf;
```

Arguments

mport

Master port implementing the inbound message unit

mbox

Inbound mailbox number

buf

Buffer to add to inbound queue

Description

Adds the *buf* buffer to the MPC85xx inbound message queue. Returns `0` on success or `-EINVAL` on failure.

Name

rio_hw_get_inb_message — Fetch inbound message from the MPC85xx message unit

Synopsis

```
void * rio_hw_get_inb_message (mport,
                               mbox);

struct rio_mport *  mport;
int                 mbox;
```

Arguments

mport

Master port implementing the inbound message unit

mbox

Inbound mailbox number

Description

Gets the next available inbound message from the inbound message queue. A pointer to the message is returned on success or NULL on failure.

Name

fsl_rio_doorbell_send — Send a MPC85xx doorbell message

Synopsis

```
int fsl_rio_doorbell_send (mport,
                           index,
                           destid,
                           data);

struct rio_mport *  mport;
int                 index;
u16                 destid;
u16                 data;
```

Arguments

mport

RapidIO master port info

index

ID of RapidIO interface

destid

Destination ID of target device

data

16-bit info field of RapidIO doorbell message

Description

Sends a MPC85xx doorbell message. Returns `0` on success or `-EINVAL` on failure.

Name

fsl_local_config_read — Generate a MPC85xx local config space read

Synopsis

```
int fsl_local_config_read (mport,
                            index,
                            offset,
                            len,
                            data);

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.

Name

fsl_local_config_write — Generate a MPC85xx local config space write

Synopsis

```
int fsl_local_config_write (mport,
                             index,
                             offset,
                             len,
                             data);

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 written

Description

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

Name

fsl_rio_config_read — Generate a MPC85xx read maintenance transaction

Synopsis

```
int fsl_rio_config_read (mport,
                         index,
                         destid,
                         hopcount,
                         offset,
                         len,
                         val);

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

Location to be read into

Description

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

Name

fsl_rio_config_write — Generate a MPC85xx write maintenance transaction

Synopsis

```
int fsl_rio_config_write (mport,
                          index,
                          destid,
                          hopcount,
                          offset,
                          len,
                          val);

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.

Name

fsl_rio_tx_handler — MPC85xx outbound message interrupt handler

Synopsis

```
irqreturn_t fsl_rio_tx_handler (irq,
                                dev_instance);

int     irq;
void *  dev_instance;
```

Arguments

irq

Linux interrupt number

dev_instance

Pointer to interrupt-specific data

Description

Handles outbound message interrupts. Executes a register outbound mailbox event handler and acks the interrupt occurrence.

Name

rio_open_outb_mbox — Initialize MPC85xx outbound mailbox

Synopsis

```
int rio_open_outb_mbox (mport,
                        dev_id,
                        mbox,
                        entries);

struct rio_mport *  mport;
void *              dev_id;
int                 mbox;
int                 entries;
```

Arguments

mport

Master port implementing the outbound message unit

dev_id

Device specific pointer to pass on event

mbox

Mailbox to open

entries

Number of entries in the outbound mailbox ring

Description

Initializes buffer ring, request the outbound message interrupt, and enables the outbound message unit. Returns `0` on success and `-EINVAL` or `-ENOMEM` on failure.

Name

rio_close_outb_mbox — Shut down MPC85xx outbound mailbox

Synopsis

```
void rio_close_outb_mbox (mport,
                          mbox);

struct rio_mport *  mport;
int                 mbox;
```

Arguments

mport

Master port implementing the outbound message unit

mbox

Mailbox to close

Description

Disables the outbound message unit, free all buffers, and frees the outbound message interrupt.

Name

fsl_rio_rx_handler — MPC85xx inbound message interrupt handler

Synopsis

```
irqreturn_t fsl_rio_rx_handler (irq,
                                dev_instance);

int     irq;
void *  dev_instance;
```

Arguments

irq

Linux interrupt number

dev_instance

Pointer to interrupt-specific data

Description

Handles inbound message interrupts. Executes a registered inbound mailbox event handler and acks the interrupt occurrence.

Name

rio_open_inb_mbox — Initialize MPC85xx inbound mailbox

Synopsis

```
int rio_open_inb_mbox (mport,
                       dev_id,
                       mbox,
                       entries);
```

```
struct rio_mport *  mport;
void *              dev_id;
int                 mbox;
int                 entries;
```

Arguments

mport

Master port implementing the inbound message unit

dev_id

Device specific pointer to pass on event

mbox

Mailbox to open

entries

Number of entries in the inbound mailbox ring

Description

Initializes buffer ring, request the inbound message interrupt, and enables the inbound message unit. Returns `0` on success and `-EINVAL` or `-ENOMEM` on failure.

Name

rio_close_inb_mbox — Shut down MPC85xx inbound mailbox

Synopsis

```
void rio_close_inb_mbox (mport,
                         mbox);

struct rio_mport *  mport;
int                 mbox;
```

Arguments

mport

Master port implementing the inbound message unit

mbox

Mailbox to close

Description

Disables the inbound message unit, free all buffers, and frees the inbound message interrupt.

Name

fsl_rio_dbell_handler — MPC85xx doorbell interrupt handler

Synopsis

```
irqreturn_t fsl_rio_dbell_handler (irq,
                                   dev_instance);
```

```
int     irq;
void *  dev_instance;
```

Arguments

irq

Linux interrupt number

dev_instance

Pointer to interrupt-specific data

Description

Handles doorbell interrupts. Parses a list of registered doorbell event handlers and executes a matching event handler.

Name

fsl_rio_doorbell_init — MPC85xx doorbell interface init

Synopsis

```
int fsl_rio_doorbell_init (mport);

struct rio_mport *  mport;
```

Arguments

mport

Master port implementing the inbound doorbell unit

Description

Initializes doorbell unit hardware and inbound DMA buffer ring. Called from `fsl_rio_setup`. Returns `0` on success or `-ENOMEM` on failure.

Name

fsl_rio_setup — Setup Freescale PowerPC RapidIO interface

Synopsis

```
int fsl_rio_setup (dev);

struct of_device *  dev;
```

Arguments

dev

of_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>