Overview

RapidIO software support consists of three interrelated source code projects/repositories:

- A character mode driver for the Linux operating system
- Ethernet frame transport over RapidIO network, known as “riosocket”
- Application (“user mode”) code which leverages the character mode driver for the following functions:
  - An application library (libmport) for the character mode driver interfaces
  - RapidIO network enumeration and management
  - The RapidIO device driver library, which provides a consistent interface for the following hardware functions: port configuration, routing table manipulation, error management, and statistics counters
  - An example file transfer application, which leverages libmport to perform RDMA style file transfer over the RapidIO network
  - The “goodput” performance measurement and debug utility, which leverages libmport to measure latency and goodput for direct I/O, DMA, and messaging.

The RapidIO software is freely available from the following public GitHub repositories:

- https://github.com/RapidIO/kernel-rapidio
- https://github.com/RapidIO/riosocket
- https://github.com/RapidIO/RapidIO_RRMAP

Note that private versions of the above repositories exist, and may be accessed by members of RapidIO.org. The private repositories contain additional functionality and features, and also contain significant test infrastructure to aid software developers.

Linux Character Mode Driver for RapidIO

RapidIO support in the Linux operating system is based on the concept of a “master port” (mport), which is a RapidIO port on an endpoint. The RapidIO port can provide direct I/O, DMA, messaging, doorbell, and other RapidIO capabilities. The RapidIO subsystem leverages these capabilities to provide the following services:

- RapidIO network enumeration and discovery for multiple mports and redundant hosts
- A sysfs file subsystem for each discovered mport and switch, exposing some attributes of each device

The character mode driver allows application software to access the direct I/O, DMA, messaging, and doorbell capabilities, if any, of the local mport(s). The libmport library, found in the RapidIO_RRMAP repository, presents the following capabilities using a standard C interface:

- Handle management – List available mports, list devices accessible from each mport, query the capabilities of each device
- Direct I/O – Map inbound and outbound windows for an mport, allowing RapidIO read and write transactions to be generated from processor reads and writes
- DMA – Perform read and write transactions over the RapidIO network
- Events – Manage event usage for an mport, and send/receive events over the RapidIO network
- Messaging – A channelized messaging (CM) capability to allow a single messaging resource to be shared using a socket-like interface

The character mode driver uses drivers for individual devices. Currently, all RapidIO switch devices are supported. The Tsi721 PCIe-to-RapidIO bridge, and some Freescale and Texas Instruments processors/DSPs, are also supported by the character mode driver.

For more information, refer to the documentation for the character mode driver found in the kernel-rapidio repository, and the wiki page for the libmport library:

https://github.com/RapidIO/RapidIO_RRMAP/wiki/2.-LIBMPORT:-RapidIO-Kernel-Driver-Library

Riosocket: Ethernet Frame Transport over RapidIO

Riosocket uses standard mport drivers to transport Ethernet frames over a RapidIO network. Riosocket is supported by mports that can send and receive DMA, Messaging, and doorbell RapidIO transactions.

Riosocket allows an mport to be managed as a standard networking device with associated MAC and IP address. Riosocket mports are managed with standard network manipulation commands (e.g. ifconfig). The MAC address is associated with the RapidIO destination ID of the device. The IP address and network mask must indicate that the RapidIO mport is connected to a distinct network. The network stack routes packets to and from the riosocket network device based on IP address. All protocols based on Internet Protocol are supported by riosocket.

Riosocket receives notifications from the RapidIO subsystem when nodes enter and exit the network. The RapidIO subsystem adds and removes nodes based on actions from application code (e.g. the RapidIO RRMAP Fabric Management Daemon). Riosocket uses a doorbell-based protocol to manage which mports are part of the Riosocket network.

Riosocket uses RDMA concepts to transport Ethernet frames. Riosocket requires a contiguous physical memory range, which must be reserved from Linux by the user. The memory range is divided into receive buffers of a fixed size. There must be one receive buffer for every other mport in the system. Each receive buffer is in turn divided into fixed size frame buffers. Short Ethernet frames are transferred with message transactions, while larger Ethernet frames are transferred with DMA writes to the frame buffers. After each DMA transfer is completed, a doorbell is sent to notify the target mport that another frame is available for processing. The maximum frame size that is transferred with message transactions can be configured, as can the number and size of receive buffers and frame buffers.

For more information on riosocket, refer to the documentation which is part of the riosocket repository.

RRMAP: Application Code and Libraries

The RapidIO Remote Memory Access Platform (RRMAP) application software consists of the following functionality:

- The libmport character mode driver library
- The RapidIO device driver library
- The Fabric Management Daemon
- File transfer example application
- Goodput performance measurement application
- Many other application libraries

Libmport

The libmport library is designed to be used by any RapidIO aware application. The libmport library can be used by applications to perform the following:

- Manage RapidIO interfaces
- Send and receive DMA transactions and doorbells
- Send and receive messages

For more information, see the libmport wiki page:

https://github.com/RapidIO/RapidIO_RRMAP/wiki/2.-LIBMPORT:-RapidIO-Kernel-Driver-Library

RapidIO Device Driver Library

The RapidIO device driver library can be used by applications to manage RapidIO devices. The device driver library interfaces present standard methods for the following functions:

- Compose and parse byte streams for control symbols and RapidIO packets
- Perform standard functions with standard registers
- Configure and query RapidIO ports
- Configure and query routing tables on RapidIO switch devices
- Configure and manage errors and other events on RapidIO devices
- Configure and read statistics counters

For more information, see the RapidIO device driver library wiki page:

https://github.com/RapidIO/RapidIO_RRMAP/wiki/7.-RapidIO-Switch-API-(LIBRIO)

Fabric Management Daemon

The Fabric Management Daemon (FMD) application is a standard way to initialize and manage a RapidIO network and the software running on the network. The (optional) configuration file allows the FMD to configure optimal routing in complex (i.e. Clos) network topologies. Additionally, the FMD allows users to assign names to RapidIO nodes based on their location in the network.

The FMD follows a master/slave design pattern. The master FMD enumerates the RapidIO network, and is the owner of all nodes in that network. All slave FMDs connect to the master FMD. The master FMD updates the slave FMDs whenever software or nodes enter or leave the network.

For more information, refer to the Fabric Management Daemon wiki page:

https://github.com/RapidIO/RapidIO_RRMAP/wiki/3.-Fabric-Management

File Transfer

The file transfer application allows files to be transferred over a RapidIO network using a customized RDMA protocol. The application follows a server/client design pattern. Files are transferred from the client to the server. Bulk data transfer is performed using DMA to memory buffers managed by the server. Coordination of the DMA transactions is performed with channelized messaging.

For more information on file transfer architecture and operation, refer to the file transfer wiki page:

https://github.com/RapidIO/RapidIO_RRMAP/wiki/4.-FILE-TRANSFER

Goodput

The “goodput” application performs latency and throughput measurements for libmport direct I/O, DMA, and messaging operations. The goodput command line interpreter manages multiple worker threads, dispatching tasks and reporting results. Scripts which perform these measurements can be automatically generated and executed.

Goodput has an extensive command set with strong automation and test capabilities. For more information, refer to the “goodput” wiki page:

https://github.com/RapidIO/RapidIO_RRMAP/wiki/5.-GOODPUT-Performance-Measurement

Revision History

Revision Date	Description of Change
June 26, 2017	Updated to encompass all RapidIO software support and new software distribution method (GitHub).
June 6, 2013	▪ Added Error Management DSF ▪ Updated Figure 2: DSF File Structure ▪ Removed support for CPS Gen1 switches from the API ▪ Completed other editorial changes
March 22 2011	Added SerDes Configuration API to DSF.
November 3, 2010	First release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.