

SEE
THE **FUTURE.**
CREATE YOUR OWN.

Embedded RTOS Support for RapidIO

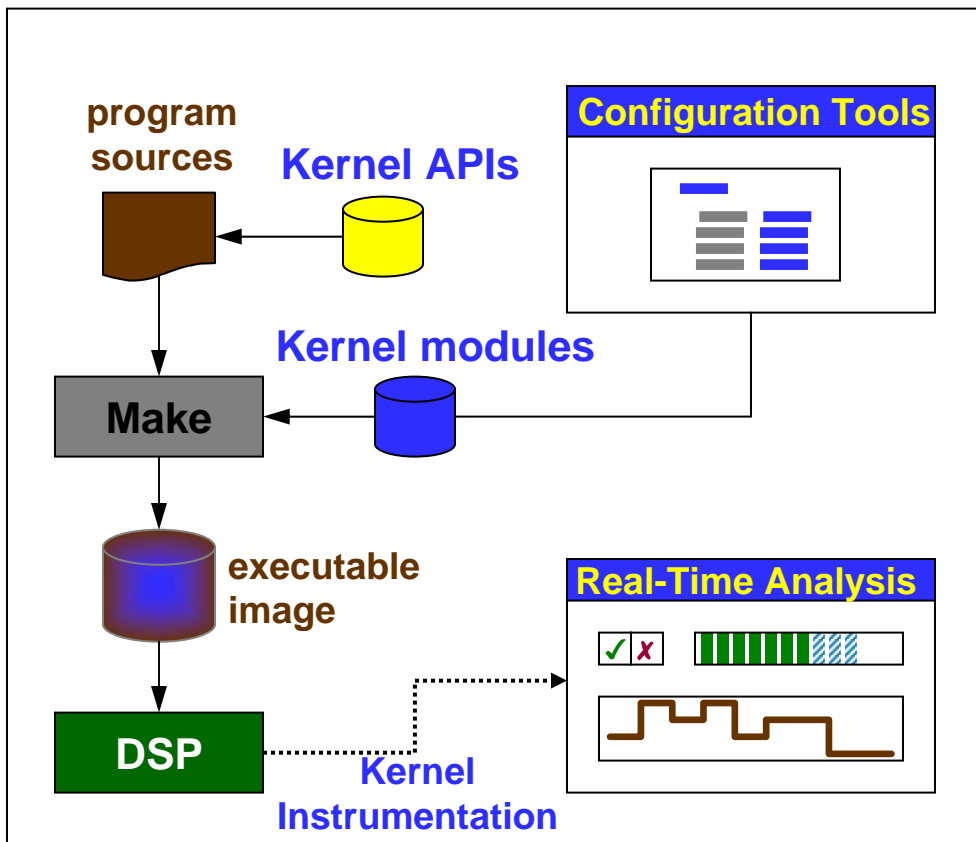


Todd Mullanix
Sr. DSP Software Eng.
Texas Instruments
Todd_mullanix@ti.com

Agenda

- ◆ **DSP/BIOS™ Kernel Overview (really short)**
- ◆ **MSGQ**
- ◆ **Transports**
- ◆ **How does RapidIO fit into all of these?**

DSP/BIOS™ Kernel Overview



- ◆ Real-Time Kernel
 - Priority Based, Preemptive Multithreading
- ◆ Configuration Tool
 - Script-based Configuration using Javascript
 - Solaris, Linux, Windows-hosted
- ◆ Real-Time Analysis Tools

Message Passing

◆ Why

- TI DSPs are being used in more complex topologies
 - In wireless handsets they must communicate with ARM
 - In basestations, multiple DSPs must communicate
- An IPC that works in both intra- and inter-processor configurations is required

◆ What

- DSP/BIOS™ Kernel Message Queue (MSGQ) module
- Message transport, notification, and allocation are abstracted from underlying system configuration
 - Supports SWI-SWI, TASK-TASK, GPP-DSP, DSP-DSP

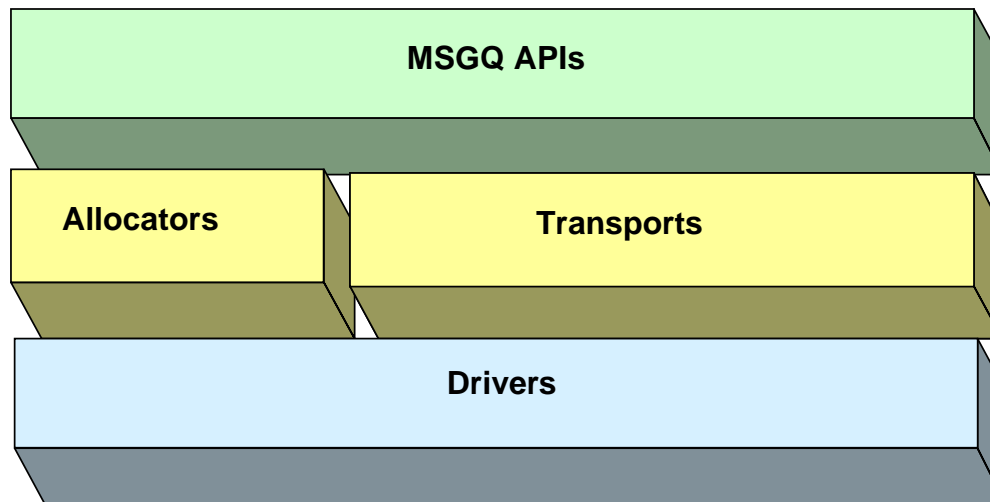
◆ When

- Available now with DSP/BIOS 5.10 and DSP/BIOS Link 1.20

MSGQ Module

The MSGQ module is comprised of three components:

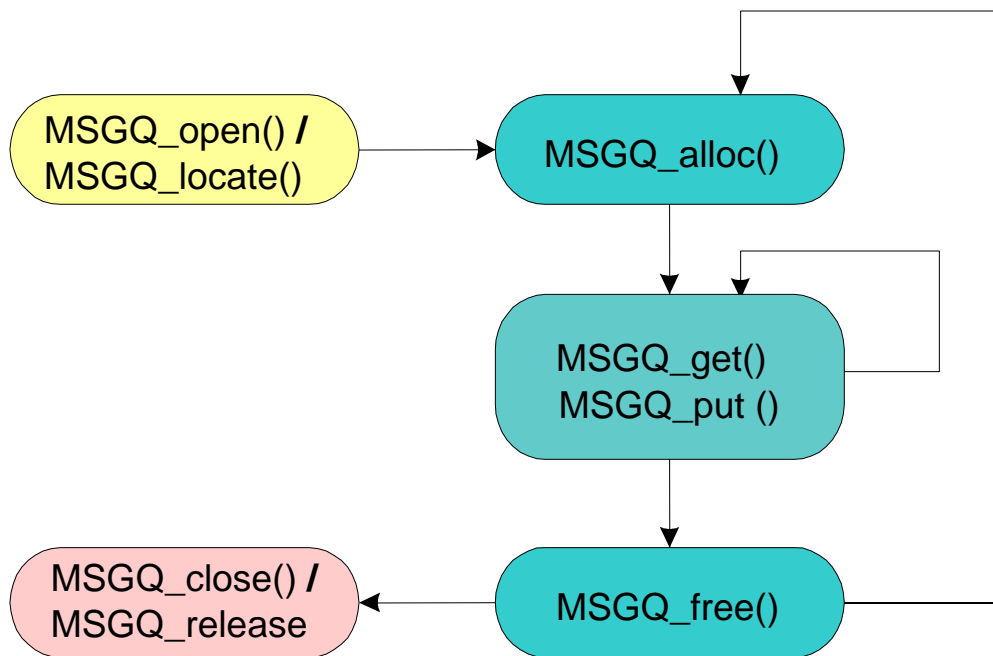
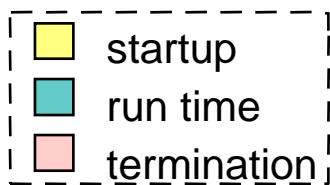
- MSGQ API: Interface that applications use. They shield the application from the transports and allocators.
- Allocators: Interface for allocating messages.
- Transports: Interface for transporting messages between processors.



MSGQ Main API Call Sequence

Reader calls:
 MSGQ_open()
 MSGQ_get()
 MSGQ_free()
 MSGQ_close()

Writer calls:
 MSGQ_locate()
 MSGQ_alloc()
 MSGQ_put()
 MSGQ_release()



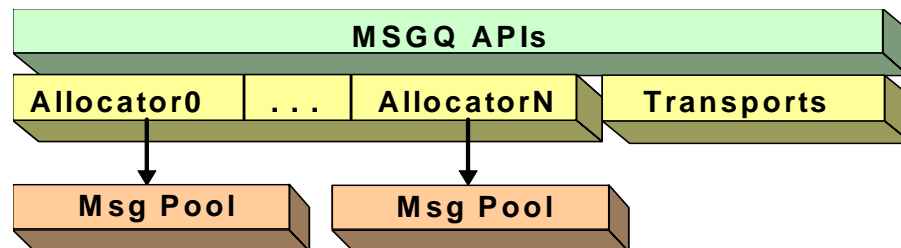
MSGQ Key Features

- **Variable length messages**
- **Zero-copy transfers of messages**
(assuming the underlying physical medium allows zero-copy in the interprocessor case)
- **Sync or async operations via application specified notification mechanisms**

The user can specify the type of notification mechanism to use per message queue (e.g. semaphore, posting of an interrupt, etc.). Thus MSGQ is not tied to a scheduling model.

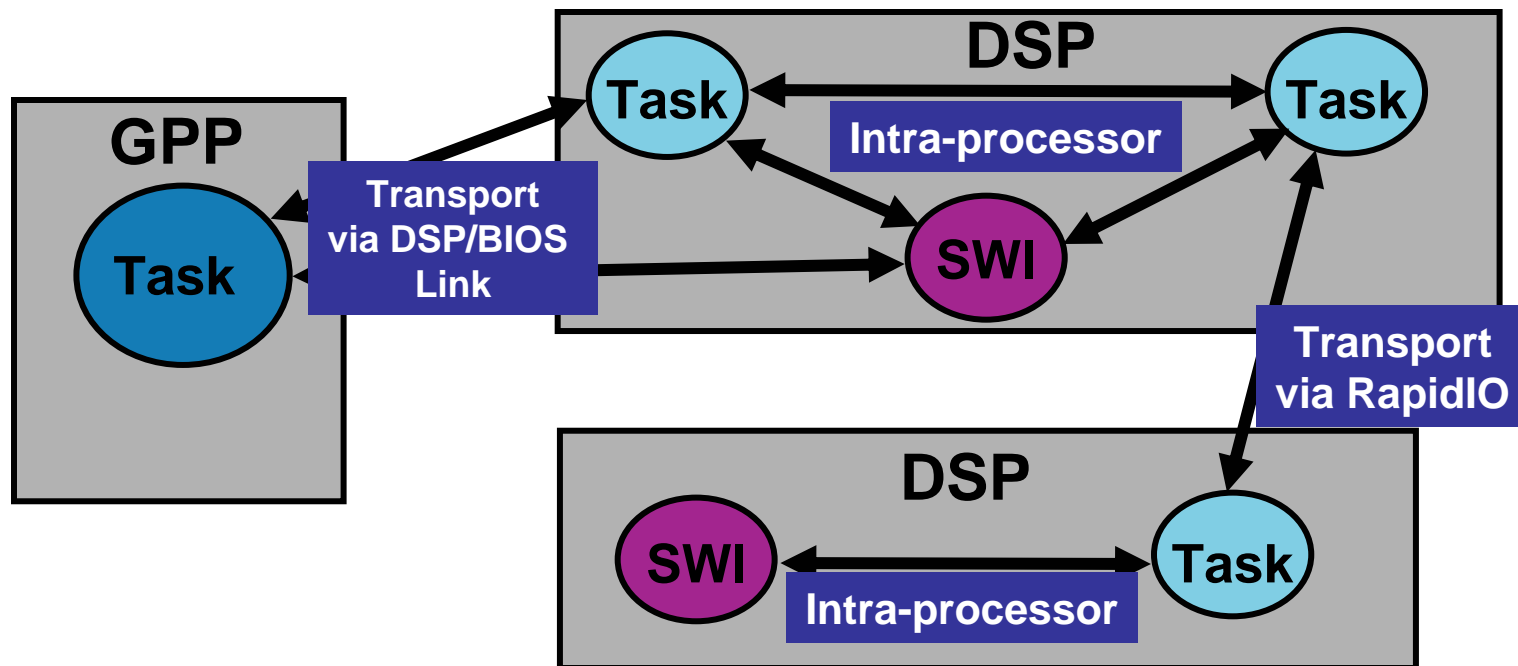
- **Quality of Service (QoS) on message allocation**

By having multiple allocators, an application can guarantee message allocation for critical resources



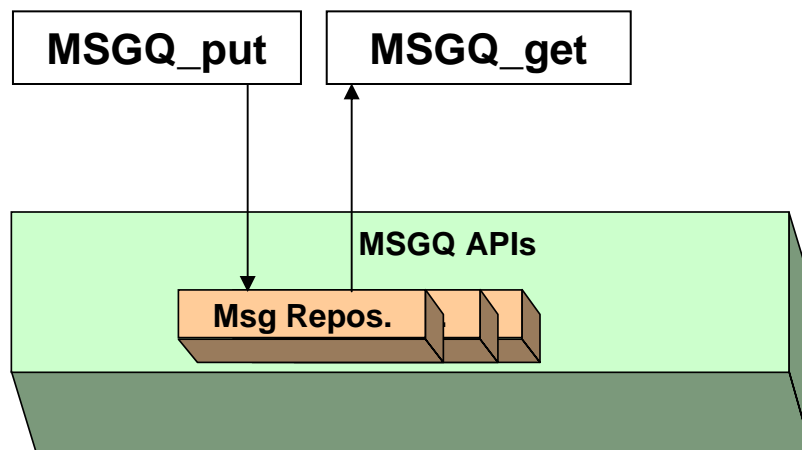
MSGQ Key Features [cont.]

- Deterministic sends and receives (assuming zero timeout on receives and the underlying physical medium is deterministic in the interprocessor case)
- Interprocessor/intraprocessor transparency
Able to move readers and writers around processors without needing to modify source code. Allows you to develop on a single processor and easily scale to a multi-processor system.



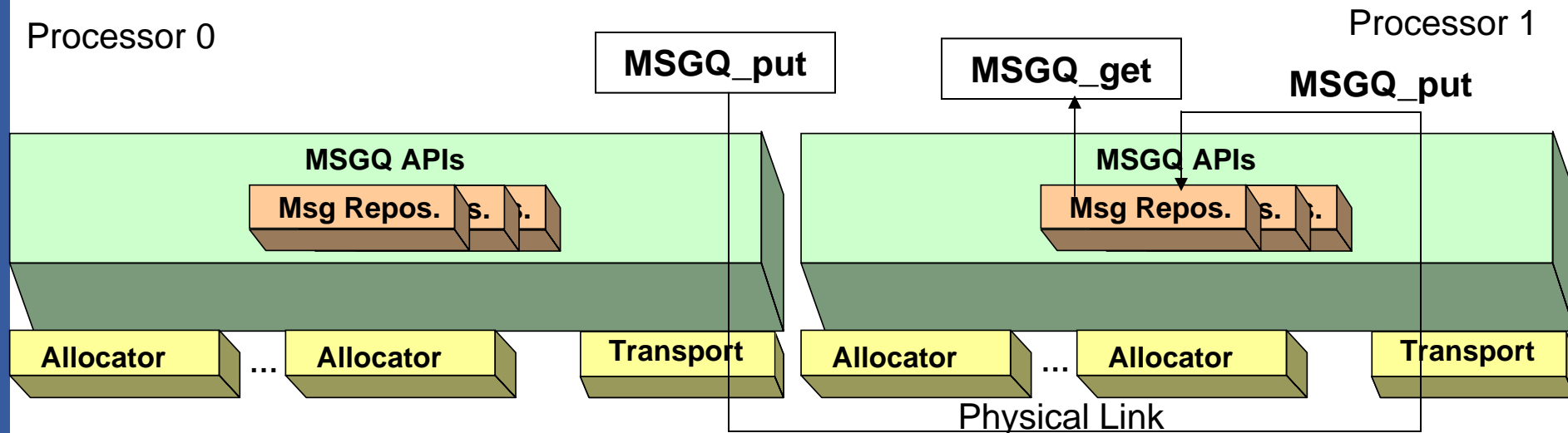
Uni-Processor

MSGQ maintains the repository of sent messages (one message repository per each opened message queue). The reader of a message queue gets messages from the message queue's repository. Note: no transports in the uni-processor case.



Transport

The job of the **transport** is to send a message across whatever physical link to the destination message queue on another processor. The transport is composed of a transport on both sides of the physical link.



By having a transport interface, an application can change the underlying communication mechanism without changing the application (except for the configuration of the transport). This approach hides the technical nuances of a physical link and allows more portability of an application.

What is a Transport

A transport is

- Library
 - Optionally, source code
- Header file
 - parameter structure
 - Name of interface function table
`extern const MSGQ_TransportFxn MQTIOM_FXNS`
 - Other required public structures

Note: The user application does not interface to the transport. This is the job of the MSGQ APIs.

RapidIO Transport for MSGQ

TI is developing a MSGQ transport for RapidIO for the c6xxx family. The package will include source code.

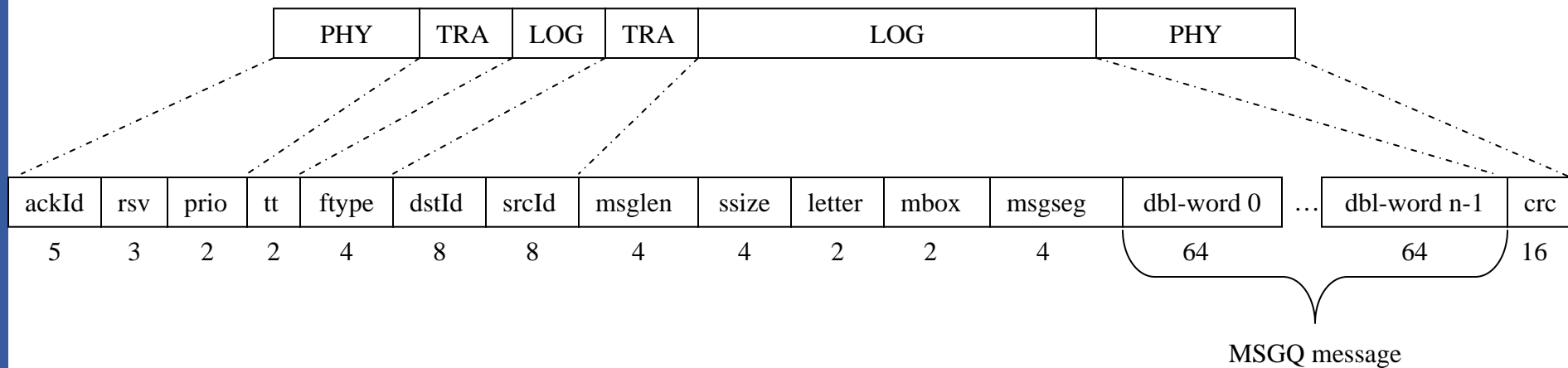
Engineers that want to use the RapidIO transport only have to add in the configuration for RapidIO transport into their applications.

Here are some of the features that this transport will support:

- Rev 1.2 Serial RapidIO Spec
- Uses the “Message Passing” Logical Specification
- (4) 1X ports configurable to a 4X Serial RapidIO port
- 1.25, 2.5 or 3.125 Gbps per diff pair
- 8b/10b Encoded data

RapidIO Segments

The following is a RapidIO Messaging segment. The MSGQ message is contained in the payload of the segment(s). The RapidIO peripheral manages the creation of the segments.

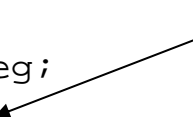


Configuration

The following are the parameters passed to each RapidIO MQT instance.

```
typedef struct RAPIDIOMQT_Params {
    /* System level parameters */
    Int      numPorts;
    Int      portSpeed;
    Int      deviceId8Bit;
    Uint16   deviceId;
    Ptr      bufDescriptorAddr;
    size_t   bufDescriptorSize;
    CSL_SrioRegs *rapidIOControlReg;
    RAPIDIOMQT_RxQueues *rxQueues;
    Int      numRxQueues;
    Int      interruptSelector;
    Int      interruptPacingValue;
    /* Instance level Tx params */
    Int      txMbox;
    Int      txCPPIQueue;
    Int      txMboxHighPri;
    Int      txCPPIQueueHighPri;
    Uint16   dstDeviceId;
    Int      numTXBuffers;
    Int      portId;
} RAPIDIOMQT_Params;
```

```
typedef struct RAPIDIOMQT_RxQueues
{
    Int      rxCPPIQueue;
    Uint16   maxMsgSize;
    Int      numRXBuffers;
    Uint16   poolId;
} RAPIDIOMQT_RxQueues;
```



Summary

You can get started today developing applications that use DSP/BIOS™ kernel's MSGQ module.

As TI rolls out devices with integrated RapidIO, you can easily migrate your applications to take advantage of this high-speed interconnect by using the TI supplied RapidIO transport for MSGQ.

Todd Mullanix
Sr. DSP Software Eng.
Texas Instruments
Todd_mullanix@ti.com

Get to market faster
with TI products,
support and partners.



SEE
THE **FUTURE.**
CREATE YOUR OWN.