20450 Century Boulevard
Germantown, MD 20874
Fax: (301) 515-7954

MSGCOM

Software Design Specification (SDS)

Revision A

QRSA000XXXX

April 11, 2014

Revision Record	
Document Title:	**Software Design Specification**
Revision	Description of Change
A	Initial Release

Note: Be sure the Revision of this document matches the QRSA record Revision letter. The revision letter increments only upon approval via the Quality Record System.

TABLE OF CONTENTS

1 Scope

This document describes the functionality, architecture, and operation of the Message communicator library.

2 References

The following references are related to the feature described in this document and shall be consulted as necessary.

No	Referenced Document	Control Number	Description
1	SYSLIB User Guide		SYSLIB User Guide
2	MSGCOM API Documentation		MSGCOM Doxygen API documentation

Table 1. Referenced Materials

3 Definitions

Acronym	**Description**
API	Application Programming Interface
DSP	Digital Signal Processor
MSGCOM	Message Communicator

Table 2. Definitions

4 MSGCOM

4.1 Introduction

The MSGCOM library is an IPC mechanism which allows messages to be exchanged between a reader and writer. The reader & writers could be located on the same DSP core, different DSP cores or even between an ARM and DSP. The goal being is that the application developers use the same MSGCOM API irrespective on how the application is broken apart.

At the lower layers the MSGCOM is based on the Navigator infrastructure which ties the library closely with the core hardware features to get the necessary performance.

4.2 Channels

MSGCOM introduces a concept called "channels" which are endpoints identified with a unique name which can be used to send and receive messages. Channels are unidirectional which are referred to as "reader channels" if the channel is capable of receiving messages. "Writer channels" are channels which are only capable of sending messages.

Reader channels are also associated with a blocking mode. The mode implies the operation of reader channel when there is no message available.

The MSGCOM module uses the named resource module to store channel name and associated information.

MSGCOM supports the following type of channels:-

1. Queue Channels
   In Queue channels messages are placed into a specific well define queue. The message memory has to be shared between the reader and writer. So fundamentally this model is similar to an IPC mechanism using shared memory but since the library is based on the Navigator hardware infrastructure atomicity across multiple cores is guaranteed by the hardware which makes the software free of any multi-core locks.

2. Queue DMA Channels
   In Queue DMA channels messages are actually transferred between the writer & reader. This provides the following benefits
   - Memory translation/paging
   - Memory independence between different SW components
   - Scattering/gathering support

3. Virtual Channels
   Due to system limitations there can only be a certain number of channels. MSGCOM provides an ability to create virtual channels over a physical channel which helps remove these limitations. Please refer to the virtual channel section below for more information.

The reader can also specify the following interrupt attributes when creating the above mentioned channels:

1. No Interrupt
   This channel does not support interrupts so basically there is no notification received by the reader when a message has been received. It is thus the responsibility of the

application to poll the channel to check if there is any message received on the specified channel.

2. Direct Interrupt
   This channel supports interrupts which implies that as soon as a writer sends a message an interrupt is immediately generated. This passes control to the MSGCOM ISR which had been plugged into the OS through the OSAL interface. Applications can thus be notified immediately with no latency on when a message is available. There are a limited number of queues in the system which support interrupts.

3. Accumulated Interrupt
   With direct interrupts mentioned above an interrupt is generated for every message which is received which under heavy load can cause an interrupt livelock issue. The Navigator infrastructure supports the ability to accumulate interrupts which will cause interrupts to be paced out. There are a limited number of accumulator queues which can be supported in the system.

All channels also support the following blocking modes:-

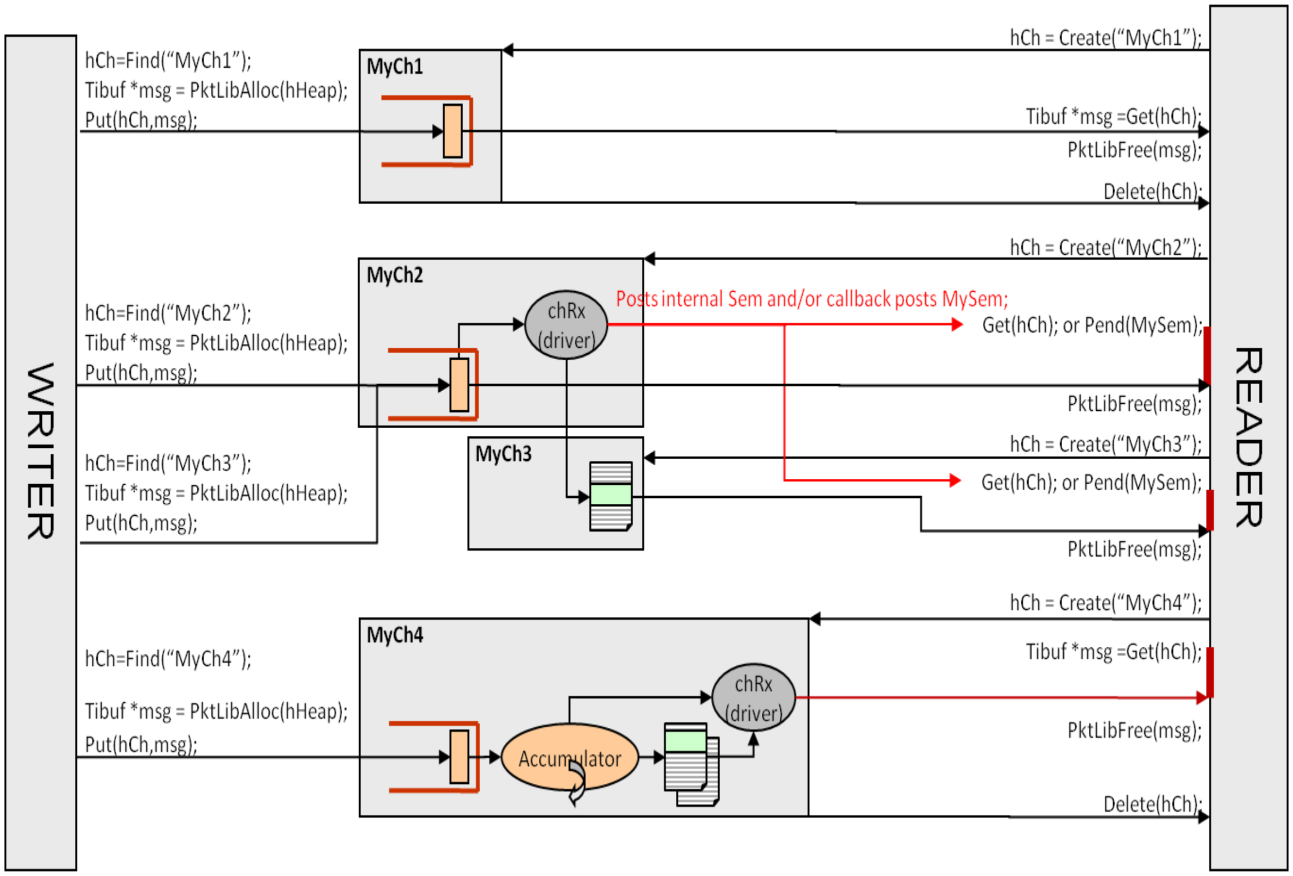
- Blocking
  In blocking mode a reader will be blocked till a message is received. The blocking mechanisms are implemented using the OSAL interface. As soon as a message is received the reader can get notified (for direct & accumulated interrupts; for non interrupt channels either a periodic poll or some other design mechanism needs to be implemented); this notification will then allow the reader to get unblocked. Thus on a blocking channel message reception will always return a received message.
- Non-blocking
  In this mode a reader will return with an empty message if there is no message available on the channel.

Another feature which is available to application developers is the ability to specify an optional call back function which is invoked every time a message is received.

The next sections document the salient features of the MSGCOM module in more detail with respect to the API usage. Please refer to the API documentation in the MSGCOM documentation directory for the latest API and also refer to the test code for usage.

4.2.1 Queue reader channels

Queue channels as explained above do not involve any copying (DMA or software). The figure below shows the various configurations in action.



Channel	Mode	Interrupt Property	Notes
MyCh1	Non-Blocking	No Interrupt	Simplest channel created using general purpose hardware queue. Reader is polling the channel, and multiple messages can be buffered.
MyCh2	Blocking	Direct Interrupt	Latency critical construction. Channel is built based on Queue with direct interrupt capabilities. Reader can be notified/awaken when message arrives.
MyCh3	Blocking (Virtual Channel)		Example of virtual channel. The amount of Queues with direct interrupt capabilities can be limited. When Application demands more latency critical channels, the virtualization technique is used. Multiple virtual channels can be linked to single Queue with direct interrupt capabilities. Both

			Reader and Writer may be unaware that they are using virtual channel instead of the physical one
MyCh4	Blocking	Accumulated	Example of accumulated channel. This construction is used to satisfy both throughput and latency requirements. HW queue with accumulation capabilities is used, providing the following benefits: - Message accumulation - Interrupt pacing - Message pre-popping

NOTE: Queue channels cannot be used to exchange data between the ARM and DSP realms.

4.2.2 Queue DMA reader channels

Queue DMA channels as explained above involve a DMA data transfer between the writer and reader. The figure below shows the various configurations in action.

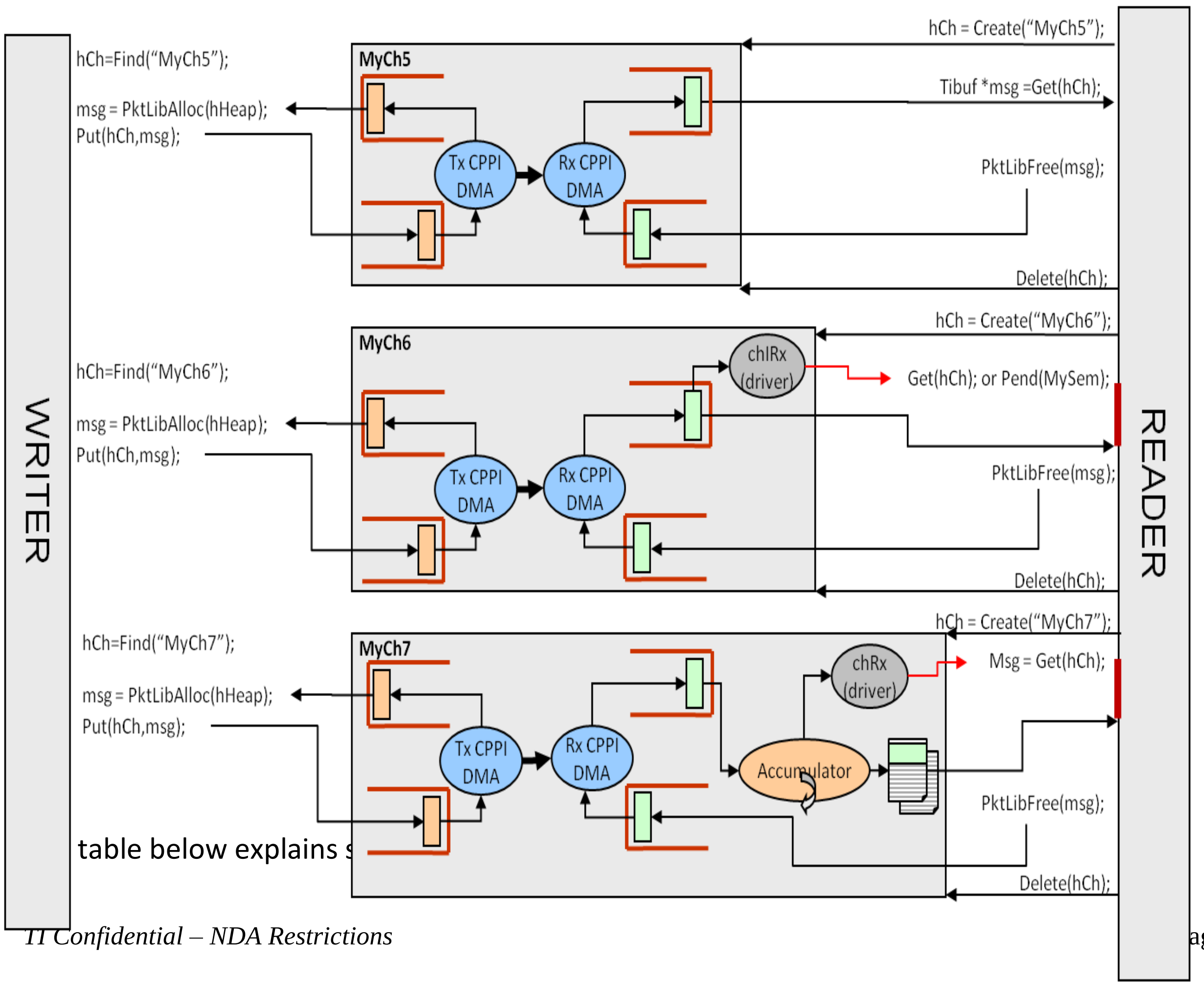


table below explains s

Channel	Mode	Interrupt Property	Notes
MyCh5	Non-Blocking	No Interrupt	Simplest channel with DMA copy support. Reader is poling the channel and multiple messages can be buffered. Similar to MyCh1 usage cases, but with memory independence and scattering/gathering support.
MyCh6	Blocking	Direct Interrupt	Similar to MyCh2, with memory independence and scattering/gathering support.
MyCh7	Blocking	Accumulated	Similar to MyCh4, with memory independence and scattering/gathering support.

Queue DMA channels as explained above involve a DMA data transfer between the writer and reader. The figure below shows the various configurations in action.

NOTE: Queue DMA channels can be used to exchange data between the ARM and DSP realms. Channel named need to be pushed between realms and this is done using the agent services.

4.3 Creating reader channels

In MSGCOM the channels are always created by the reader core. Please refer to the following MSGCOM API: `Msgcom_create`

While creating the channel the configuration properties described above are specified. Please refer to the `Msgcom_ChCfg` for more information. Once a channel is created it can now be used to receive messages from the remote endpoint. Each channel is identified by a unique name.

4.4 Creating writer channel

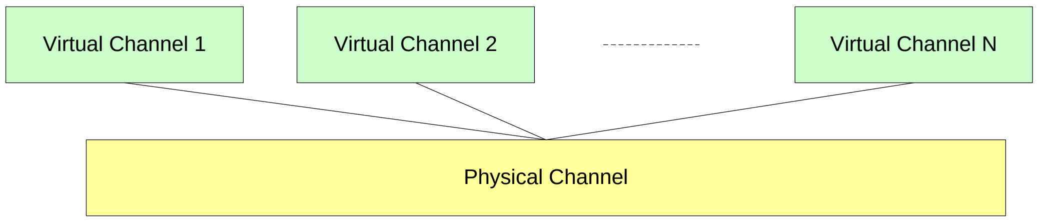
As specified above; channels can only be created by the "reader". Channels are registered and placed into an internal synchronized database which is accessible across the system. Writers can thus find a channel on which they wish to communicate with the reader using the following API: `Msgcom_find`.

The function returns the channel handle. The writer can then use this channel handle to send messages to the remote endpoint.

4.5 Virtual Channels

As mentioned above; certain channels such as direct & accumulated interrupt channels have system hardware limitations. These channels can be very useful and the hardware limitations might not scale to the application requirements. To mitigate this issue the MSGCOM introduces a concept called virtual channels.

The figure below shows the architecture of this concept:-



There can be 32 virtual channels created on top of a physical channel. Virtual channels like physical channels are only created on the reader core. Internally each virtual channel is associated a unique identifier. When a message is sent this virtual identifier is passed as a part of the message. The reader core on the reception of the message will decode the identifier to determine the virtual channel on which the message is received.

Each virtual channel is associated with a software receive buffer (which will hold a list of all received packets). The size of this software buffer is configurable and specified during virtual channel creation. Once the received message is taken out of the physical channel it is then placed into this receive buffer list.

While creating the MSGCOM virtual channel the configuration requires the specification of a physical channel over which the virtual channel reside. Once the physical channel has been *virtualized*; the message send and receive function will no longer operate on the physical channel. The virtual channel will inherit the properties of the physical channel. *For example:* If the physical channel was a direct interrupt blocking mode channel then all virtual channels would have the same behavior.

4.6 Sending message

Messages are sent only by the writer core using the following API: `Msgcom_putMessage`. The channel handle as mentioned above can be found on the writer by performing a find operation. The message buffer is a pointer to the message which is to be sent and this should be a pointer to the PKTLIB data structure.

Cache coherency on the data buffer should be handled by the application. However cache coherency on the PKTLIB data structure is handled by the MSGCOM module.

4.7 Receiving message

Messages are received only by the reader core using the following API: `Msgcom_getMessage`. The channel handle indicates the channel on which the message is to be received. If a message is received the pointer to the received message is passed back in the message buffer.

Cache coherency on the data buffer should be handled by the application. However cache coherency on the PKTLIB data structure is handled by the MSGCOM module.

4.8 Delete Channel

The MSGCOM channel can be deleted using the following API: `Msgcom_delete`. The API will delete the channel and will use the provided function to cleanup messages which are still pending to be serviced in the internal MSGCOM receive & transmit queues. There are certain rules which need to be followed while calling this API

- Readers and writers need to delete their own MSGCOM channel handle.
- The application is responsible for ensuring that the MSGCOM channel is not being used in the reader & writer during & after deletion. The MSGCOM API does not perform any run time checking to ensure that the MSGCOM channel is valid or not because this will have a performance penalty.
- Physical channels which are virtualized; can only be deleted once all the virtual channels associated with the physical channel have been deleted.