Ferry Bridge

Programmers Guide

August, 2017

Revision 0.6

Intel Confidential

Document Number: XXXXXX-0.x

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

This document contains information on products in the design phase of development.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

About This Document

Purpose

This document describes the Software Architecture of the Ferry Bridge FPGA and is complimentary to the LTE L1 Reference Stack Software documentation. This document is another addition to the extensive Software documentation listed in below section and is intended to brief the reader on the usage models for the FPGA

Intended Audience

This document is intended to be used by engineers and architects who wish to develop an understanding of the Ferry Bridge Software, the workloads and/or services supported by the FPGA.

Companion Documents

A number of key companion documents are identified.

- Ferry Bridge User Guide
- Ferry Bridge Release Notes

Contents

List Figures

List of Tables

Revision History

Revision Number	Description	Date
0.5	Initial release (headings)	August 2016
0.6	Updates for R1.4.0 release	August 2017

§

1 *Introduction*

The Ferry Bridge FPGA is targeted as an enabling device supporting the Common Public Radio Interface (CPRI), pre-processing for LTE Layer 1 (PHY) protocols and an Ethernet back-haul through a "Front End" architecture for Cloud Radio Access.

1.1 Scope

The scope of this document includes the Ferry Bridge FPGA and the deliveries of software associated with the FPGA. Other software components are described where necessary to provide the reader with a complete view of the FPGA software. An example of another component is the Baseband software stacks or the Intel Data Plane Development Kit.

1.2 Terminology

Term	Description
FB	Ferry Bridge
RNL	Radio Network Layer
TNL	Transport Network Layer
VM	Virtual Machine
CPRI	Common Public Radio Interface
RT	Real-time
RB	Resource Blocks
RE	Resource Elements

1.3 Reference Documents

Document	Document No./Location
Landing Zone	
Product Requirements Document (PRD)	0.7 30th April 2013
External Architecture Specification	28979/0.91
LTE Multi-Sector Reference Architecture	0.3
WCK ASIC Register Mapping Block MAS	0.52 04-Feb-2013
Building a Linux* Based Real-Time IA Platform in Native and Virtualized Environments – Tools, Techniques and Recommendations	514108
Ferry Bridge Release Notes	
Ferry Bridge User Guide	

§

2 Assumptions, Dependencies and Constraints

2.1 Assumptions

Device Configuration will be performed by the Ferry Bridge library.

The implementation of the LTE SW stack is outside the scope of this document.

2.2 Dependencies

Intentionally blank.

2.3 Constraints

The single usage mode applies to all the CPRI links presented by a single instance of the Ferry Bridge driver.

A platform will be connected to a single Front End device.

§

3 *Architecture Overview*

3.1 Introduction

3.1.1 Wireless Software Protocol Stacks

A brief summary of the wireless protocols running on an eNodeB is provided in this section. The interaction between the protocol layers and the fixed cell processing and IO by Ferry Bridge FPGA is described.

Note: A description of the protocols typically running on an eNodeB is outside the scope of this document.

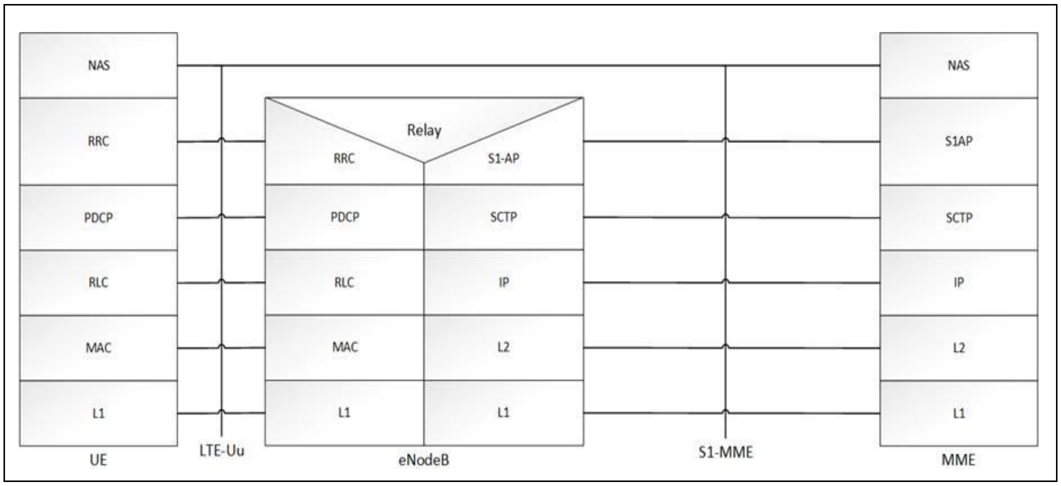
A summary of the protocols specified by 3GPP for an eNodeB is listed.

- Transport Network Layer/backhaul interface: S1-MME (MME), S1-U (SGW), X2. (eNodeB to eNodeB).
- Layer 3 protocols: RRC
- Layer 2 protocols: MAC/RLC/PDCP
    - In LTE, PDCP is responsible for ciphering and integrity calculation for radio bearers. No accelerations are provided for the L2 protocols.
- Layer 1 protocol: PHY and CPRI to RRH
    - The PHY or physical layer carries information from the transport channels over the Air interface. In this case the IO is handled through the Ferry Bridge FPGA connected over Ethernet.

The Ferry Bridge FPGA provides fixed cell processing for Layer 1 (PHY) and CPRI IO.

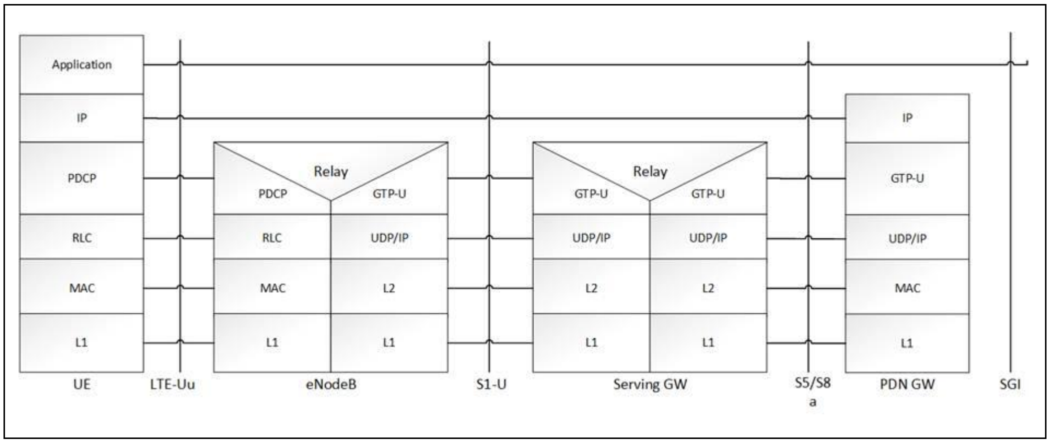
The protocol stack layers on the control plane path including the eNodeB and its interfaces are shown for reference, control plane traffic flows on this interface.

Figure 3-1. Control Plane Protocol Stack



The protocol stack layers on the user plane path including the eNodeB and its interfaces are shown for reference, user plane traffic flows on this interface.

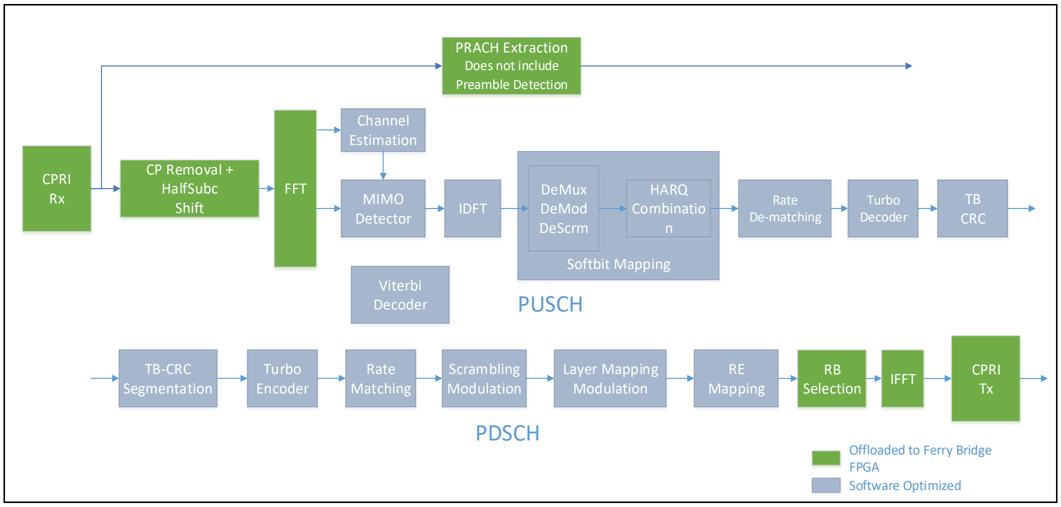
Figure 3-2. User Plane Protocol Stack



3.1.2 Protocol Layer Mapping to Front End

The diagram provided shows the Ferry Bridge in context of the LTE PHY protocol layers and software elements.

Figure 3-3. Protocol Mapping to Ferry Bridge Fixed Cell Accelerations

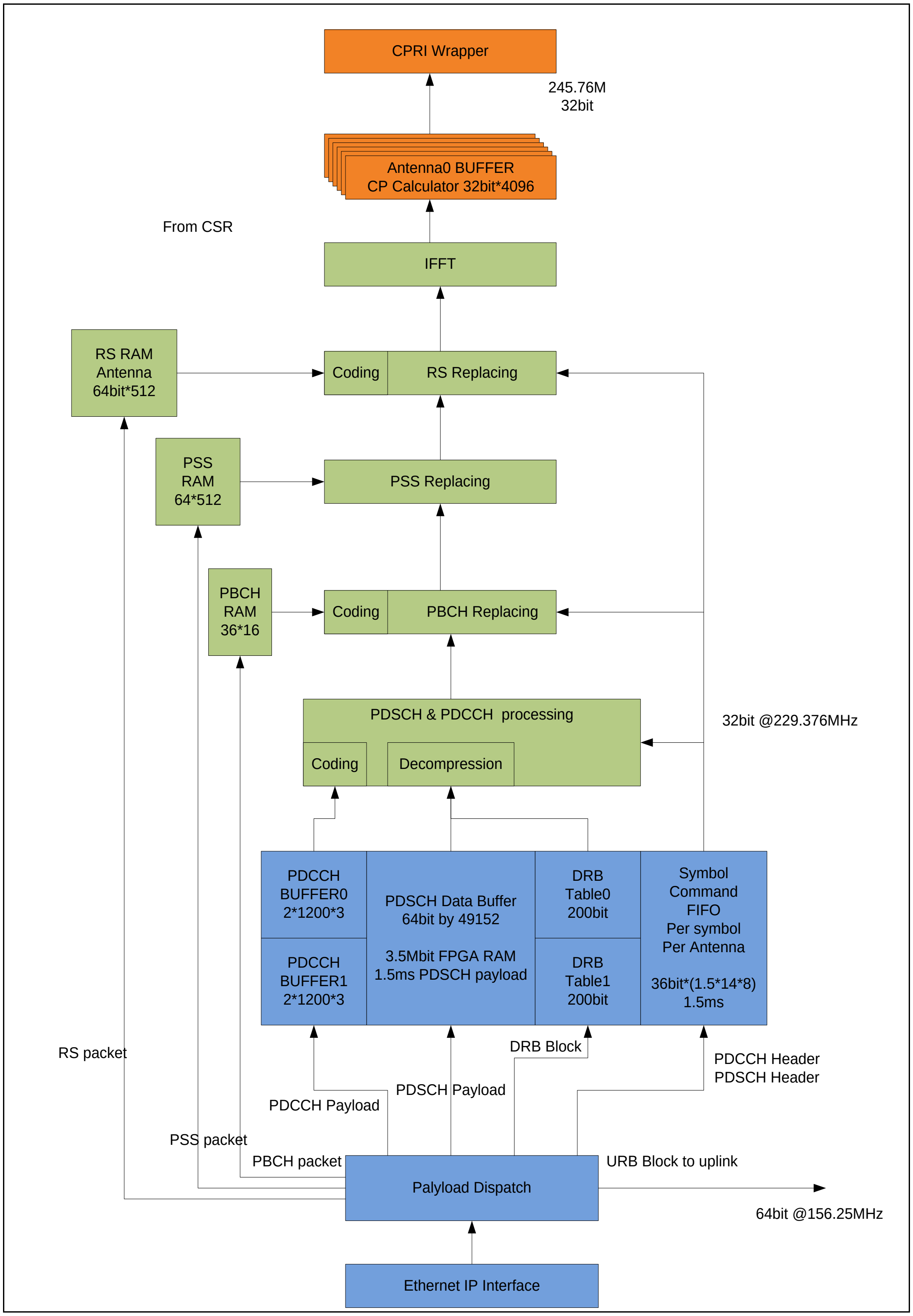


Note: Compression function is not show in Figure 3-3 as it is not a protocol function, but it is implemented in the Ferry Bridge FPGA and Software.

The mapping show in Figure 3-3 is specific to Ferry Bridge FPGA and not all features/functions may be implemented in the first release. The communication between the SW and the Ferry Bridge FPGA is not shown in this diagram, it is discussed later section. Other LTE channels/processing provided but not shown in Figure 3-3 include PDCCH, PBCH, PSS and RS processing.

Figure 3-4 shows the high level block diagram of the HW functionality. This shows the mapping of the additional control channels and block mapping.

Figure 3-4. HW High Level Block Diagram Downlink



3.2 Optimizations for Real-Time

The Ferry Bridge software requires the system to be configured for real time, please see Comprehensive guide to configuring a system for Real-time is included in document number 514108.

3.3 Hardware Context

In the first phase of the Ferry Bridge program, the FPGA will be a single device and present a single interface to the Software. The interface will be accessed over Ethernet and will contain a single Front End capable of performing the fixed cell processing like FFT for a single sector/cell of LTE L1.

3.3.1 Virtualization and SR-IOV

As the Ferry Bridge Driver uses the DPDK user space driver directly, it assumes all the Virtualization features from DPDK. It is expected that the Ferry Bridge lib will have direct control over the Ethernet port that is being used to communicate to the Ferry Bridge FPGA. There will be no explicit Virtualization technologies implemented in the Ferry Bridge lib.

3.4 Software Context

The Ferry Bridge driver is different from most device drivers as the physical hardware that it controls is not local to the Intel Server platform that the software is executing on. The physical hardware (FPGA) is connected over Ethernet to the Server, as such all communication between the Ferry Bridge driver and library must be over Ethernet. Figure 3-5 shows present a high level overview of the software and hardware configuration, where the Ferry Bridge software is communicating with the Ferry Bridge Hardware over Ethernet. The Ferry Bridge Driver shall utilize the DPDK infrastructure to send and receive Ethernet packets directly to the Front End FPGA.

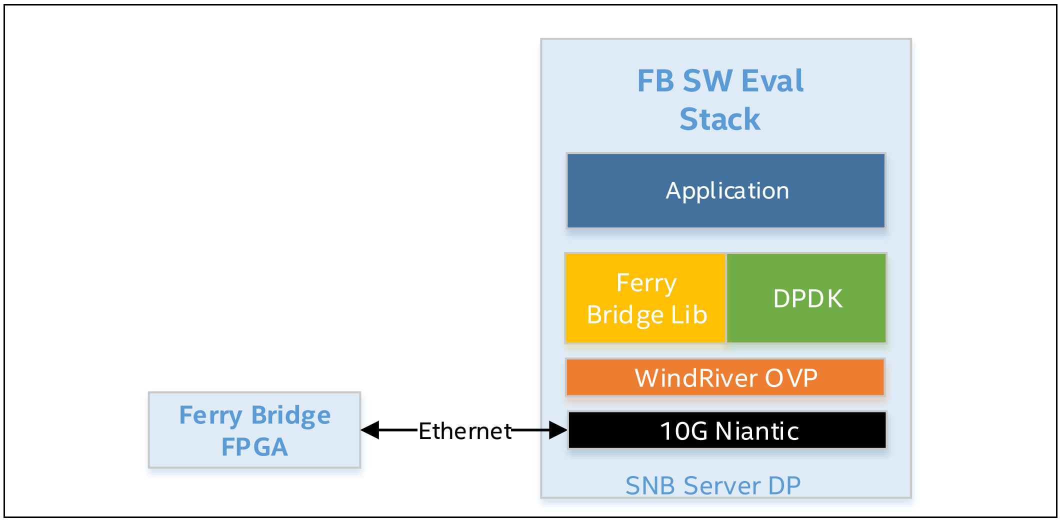
3.4.1 Host Software Components

This section gives an introduction to the software components in a non-virtualized/Host only environment.

- Header files and a library implementing the Ferry Bridge APIs is presented to the user space application.
    - An application may link this library statically or dynamically. For Phase 1 only static linking will be supported.
    - The Ferry Bridge Library will present configuration and control API's to the application.

Figure 3-5 shows a high level overview of the SW context for Ferry Bridge. The application will use Ferry Bridge Library to initialize and configure the Front End device, initiating the IO links. The Ferry Bridge Library will utilize DPDK receive and transmit queues to communicate with the Front End device. It is the responsibility of the application to correctly configure DPDK for the communication. There are two flows through the system, one for data path and one for control path. Both flows will utilize DPDK.

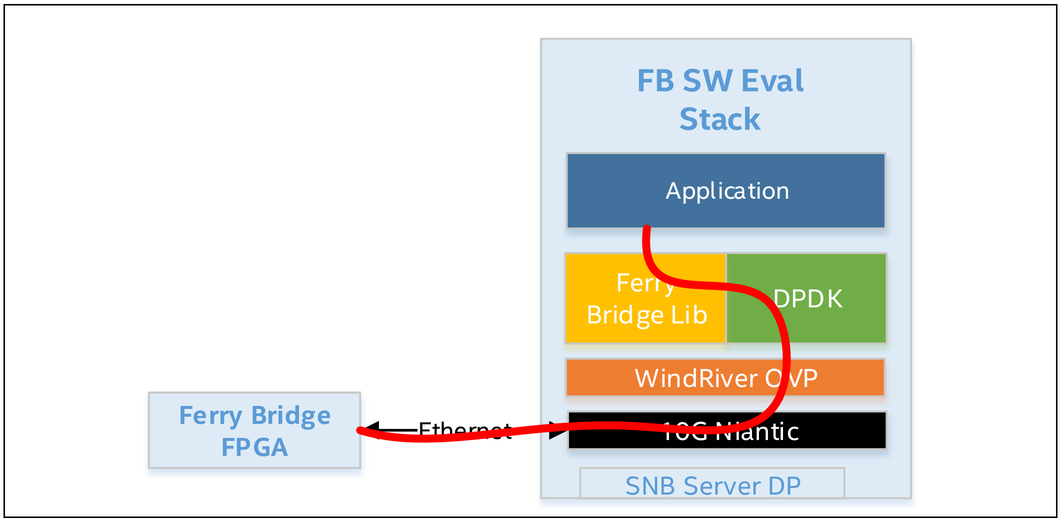
Figure 3-5. High Level Software Overview Host



3.4.1.1 Ferry Bridge Control Path Flow

The control path flow will be used for all initialization and configuration options of the Front End device. The application will invoke the Ferry Bridge Library via an API call. The Ferry Bridge library will then initiate communication with the Front End Device and will start to transmit and receive control messages. Once the Ferry Bridge library has been invoked, it will assume ownership of the DPDK Rx and Tx queues and will only release once the configuration operation is complete. To prevent blocking of the Data Path, VLAN Tags will be used to classify the Control Path Packets onto a separate queue so that data path packets can still be received on the data queue.

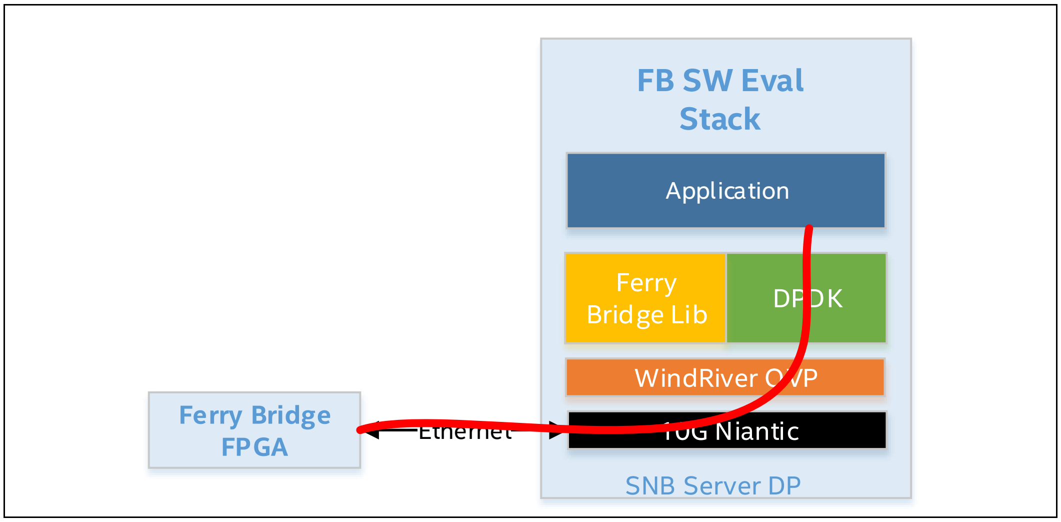
Figure 3-6. Ferry Bridge Control Path Flow



3.4.1.2 Ferry Bridge Data Path Flow

The data path flow through the system will consist of data packets send directly by the application to the Front End Device, using DPDK. The Application is responsible for forming the packets with the correct header and ensure the payload is in the correct format as expected by the Front End device.

Figure 3-7. Ferry Bridge Data Path Flow



3.4.2 Virtualization - Software component overview

This section provides an overview of the software components in a virtualized environment for a single instance of a Virtual Machine. Multiple instances of the virtual machine and the drivers shown will operate in a typical virtualized platform.

3.4.3 Packet Classification

Packet Classification will be done in both Hardware and Software. There are limitations in doing packet classification in SW only, this is because the Ferry Bridge API's which use predefined DPDK queue's to communicate with the Remote Device, assume ownership of the queues for the duration of API implementation. This implies that if no HW classification is done, all packets received on the DPDK queues while an API is executing and that are not intended responses to the API will be dropped. Because of this VLAN Tags will be used to classify ingress packets originating from the Front End Device. Packets will be classified based on the VLAN Tag and DCB for priority. The Application will be responsible for correctly setting up the VLAN Tag filter tables to ensure packets are correctly classified and placed on the correct queue. This allows the application to perform a form of hardware packet classification. In the first release this will primarily be used to classify control packets from data packets. In separating onto separate queues it allows the Ferry Bridge library to take ownership of a control queue and wait for the packet responses from the remote device without interfering with the data packets which will be separated onto a different queue in the NIC. In future releases it will be possible to configure the VLAN tags to separate the different data packets PUSCH, PRACH etc.

3.4.4 Real-time partitioning cores

To achieve real-time a specific configuration is required to partition the cores with real-time workloads versus those without. This section discusses the constraints applied.to meet the real-time needs of the platform.

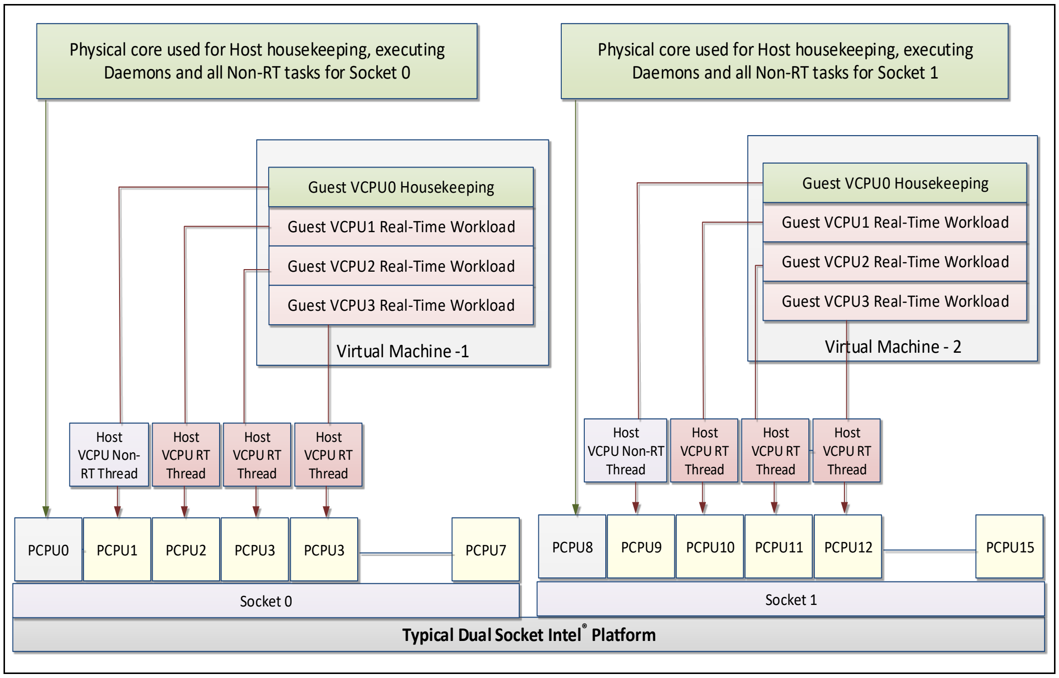
3.4.4.1 Guest Side CPU Resource Partitioning

On the guest side, it is required to have one VCPU per guest (minimum) that is not configured for real-time and runs all non-real-time workload and housekeeping tasks of the guest. All other VCPUs of a guest are isolated from non-real-time workloads and are reserved for the real-time workload.

CPU usage by the workload is a critical factor that determines the VCPU to PCPU mapping. For example, if the real-time workload requires 80% to 90% of the CPU resource, then each guest VCPU that is running a real-time workload is pinned to one of the reserved physical cores on the host. It is only possible to have one real-time VCPU per physical core. If the CPU usage is relatively low and the real-time workload is designed in such a way that real-time workloads running on multiple VCPU can co-operatively schedule between each other, then it may be possible to pin multiple VPCUs running real-time workloads onto a single PCPU.

The diagram below shows an example CPU resource allocation in a virtualized environment on a dual socket platform. This is the setup used for all the benchmarks presented in this application note.

Figure 3-8. Typical Virtual CPU to Physical Mapping


Physical core used for Host housekeeping, executing Daemons and all Non-RT tasks for Socket 0
Physical core used for Host housekeeping, executing Daemons and all Non-RT tasks for Socket 1
Guest VCPU0 Housekeeping
Guest VCPU1 Real-Time Workload
Guest VCPU2 Real-Time Workload
Guest VCPU3 Real-Time Workload
Virtual Machine -1
Guest VCPU0 Housekeeping
Guest VCPU1 Real-Time Workload
Guest VCPU2 Real-Time Workload
Guest VCPU3 Real-Time Workload
Virtual Machine - 2
Host VCPU Non-RT Thread
Host VCPU RT Thread
Host VCPU RT Thread
Host VCPU RT Thread
Host VCPU Non-RT Thread
Host VCPU RT Thread
Host VCPU RT Thread
Host VCPU RT Thread
PCPU0
PCPU1
PCPU2
PCPU3
PCPU3
PCPU7
PCPU8
PCPU9
PCPU10
PCPU11
PCPU12
PCPU15
Socket 0
Socket 1
Typical Dual Socket Intel® Platform

As indicated earlier optimization consideration for consolidating housing VCPUs of all the VMs on the same socket on a single physical core needs to be investigated.

For multi-socket systems, there is an additional restriction: A performance limiting factor, guests are restricted from spawning their VCPUs and their host side hypervisor threads across sockets. Thus, avoiding high latency cross socket memory access over the QPI bus.

3.5 Threading Model

As the Ferry Bridge library is responsible for the control and configuration of the Remote Device only it is not envisaged that multiple threads will be needed as such the Ferry Bridge library will be implemented as a single threaded design, utilizing the application calling thread context. The data path communication is not implemented in the Ferry Bridge library, it is accessed directly by the application itself, as defined in 3.4.1.2. In the first phase of project a single queue will be used for all data path communication with SW classifying the different data packets (PUSCH, PRACH, Timing Packets etc.). As a result the use of a single DPDK queue implies a single thread for communicating with the Ferry Bridge remote device.

The overall threading model used by the application is not in the scope of this document.

3.6 Buffer/Memory Management

Buffer and memory management will be handled by the application, and not the Ferry Bridge library. It is expected that all buffers passed to the Ferry Bridge library will be DPDK MBUF buffers allocated from the DPDK buffer and memory manager.

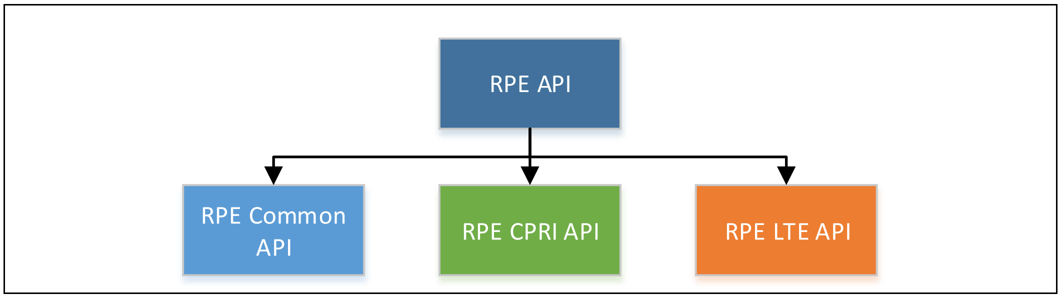
4 Ferry Bridge Interface Definition

As Ferry Bridge is a remote device, there are two distinct interface that will be defined, these are the API definition for interfacing with the SW library and the Ethernet packet definition for interfacing with the Ferry Bridge remote device directly. These interface definitions will both be defined in RPE Ferry Bridge API which can be found in Reference Documents. Ferry Bridge API structure is below.

4.1 RPE Ferry Bridge API Structure

The Remote Processing Engine (RPE) API definition is divided into 3 distinct components as outlined in Figure 4-1.

Figure 4-1. RPE Ferry Bridge API Structure



4.1.1 RPE API

The top level API definition for the Remote Processing Engine. It defines structures and data types that are common across the interface. The software application will be presented with an instance handle. The Instance handle will be used by the application to interface with a particular instance of the RPE Device. That instance will present different functionality such as CPRI connectivity and LTE pre-processing, which can be configured through the various APIs (RPE CPRI API, and RPE LTE API). In the case of multiple sectors, the application will be presented with multiple instance handles, each instance handle is mapped directly to a single eNodeB sector/cell. Note that multiple instance handles can map to the same RPE device.

4.1.2 RPE Common API

Common API provides common function definitions, such as initialization and configuration options for the RPE API.

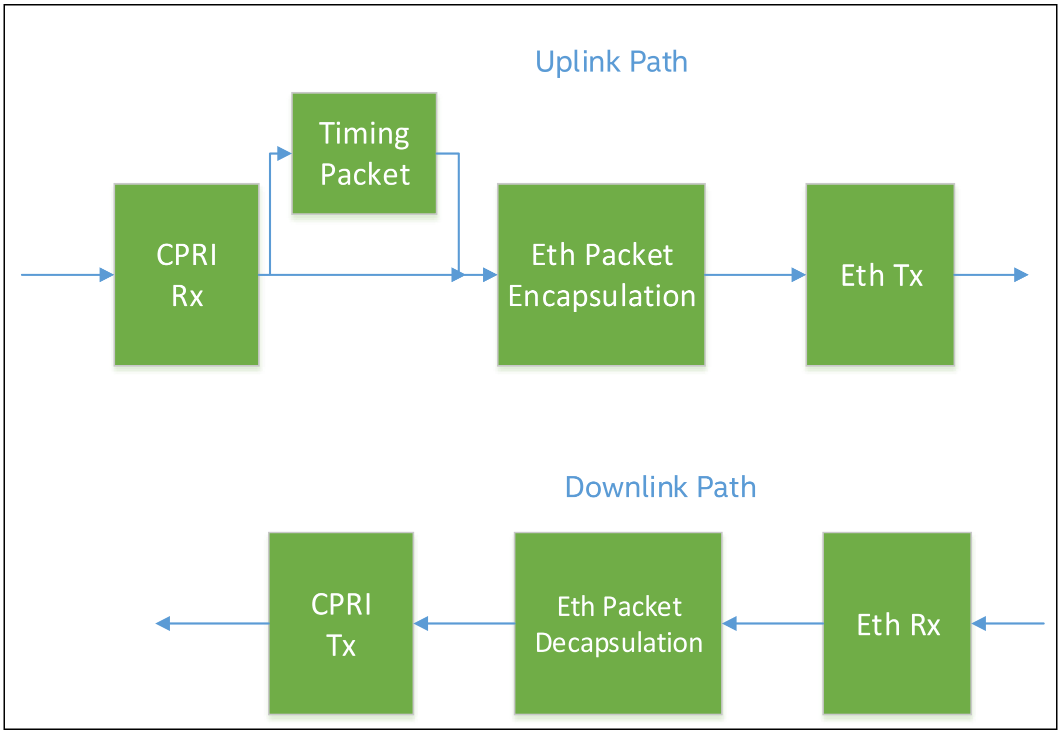
4.1.3 RPE CPRI API

The RPE CPRI API defines functions and structures that can be used to configure and setup a CPRI instance associated with the Remote Processing Engine. These API's are for control and setup only.

4.1.3.1 RPE CPRI Packet API

The RPE CPRI Packet API defines the packet structure that should be used when communicating with the RPE device. It includes both the control packet structure that is utilized by the library itself and also the CPRI data packet structure which can be used directly by the application when communicating directly with the Remote Processing Engine. The CPRI Packets are used when the RPE is configured for By-Pass mode. In this mode all the LTE processing is bypassed and the Ethernet Packets are mapped directly to the CPRI interface. This allows Software to communicate directly over CPRI.

Figure 4-2. RPE CPRI Mode (Bypass)



4.1.4 RPE LTE API

The RPE LTE API defines functions and structures that can be used to configure the LTE specific features associated with the remote processing engine. These API's are for control and setup only.

4.1.4.1 RPE LTE Packet API

The RPE LTE Packet API defines the packet structure that should be used when communicating with the LTE processing instance in the RPE device. It includes both the control packet structure that is utilized by the library itself and also the LTE data packet structure which can be used directly by the application.

4.2 High Level RPE Packet Definition

Note: All packets are defined in Network Byte Order and will need to be converted to little endian for SW definition.

The Ethernet Packet structures to be used are defined fully in the in RPE Ferry Bridge API which can be found in Reference Documents. This section gives an overview of the packet structure and the packet types and the associated VLAN tags that will be used to filter the packets on the uplink path. Packets are divided into two different flows, upstream flow (packets sent from the RPE to the IA server and received by DPDK) and downstream flow (packets sent from the IA server through DPDK and received by the RPE).

There are two types of packets, control and data.

- Control
    - General control, configuration space access and status indicators sent from RPE.
- Data
    - Data to be processed by LTE functions and sent over CPRI interface. Can be further sub-categorized depending on the functionality enabled in the RPE.

As VLAN Tags shall be used for packet classification, each packet shall contain the same Ethernet Header structure that complies with VLAN tags. The Ethernet Type will always be set as defined in DPDK for VLAN TAGs see Code Snippet 1.

Code Snippet 1. Ethernet TAG Protocol ID Definition

```
/* Extract from  rte_ether.h */

#define ETHER_TYPE_VLAN 0x8100 /**< IEEE 802.1Q VLAN tagging. */
```

All packets shall follow the high level packet structure as outline in Figure 4-3.

Figure 4-3. High Level Packet Structure showing VLAN TAG

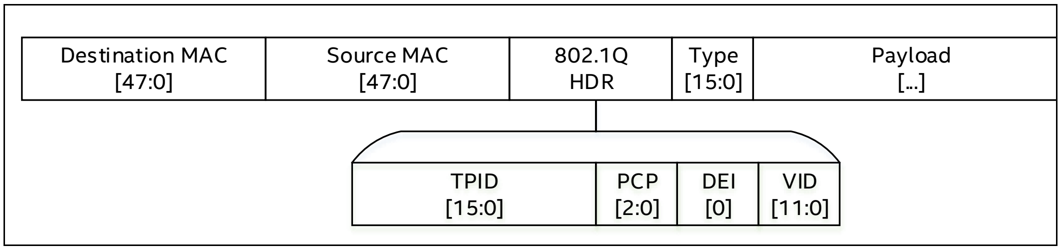


Table 4.1. VLAN TAG Field descriptions

Field	Description
TPID	Tag Protocol Identification. Should be set to 0x8100 to indicate VLAN TAG. 16 bit field.
PCP	3 bit field indicating the priority of the packet. Highest priority is 7, lowest priority is 0
DEI	Drop Eligible indicator a 1 bit field to indicate if frames can be dropped.
VID	VLAN Identification 12 bit field.

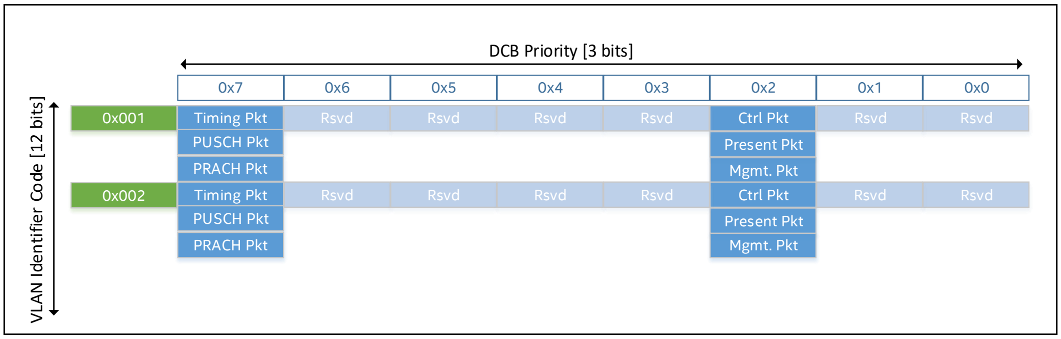
As discussed in 4.2.1 VLAN Tag Configuration, the VLAN Tag will be used to identify the CPRI Link and the Priority PCP field will be used to identify the Packet Type used.

4.2.1 VLAN Tag configuration

Current VLAN tag implementation allows filtering based on VLAN tag and DCB priority. Using both these features allows SW to filter on by both CPRI Link and the type of packet being received. Current network driver allows has a max 128 VMD Queues that can be used for this filter. These must be used in a certain fixed configurations one of which is 16x8, this means there are 16 queues to filter on VLAN Tag type and 8 queues to filter on the DCB priority. The VLAN Tags will be mapped to CPRI Link ID's and the DCB priorities will be mapped to the packet Type.

Note: In the first phase the Data packets will not be separated onto separate HW queues and the software application itself will perform the packet classifications.

Figure 4-4. VLAN Tag mapping to CPRI Link and Packet Type



	0x7	0x6	0x5	0x4	0x3	0x2	0x1	0x0
0x001	Timing Pkt	Rsvd	Rsvd	Rsvd	Rsvd	Ctrl Pkt	Rsvd	Rsvd
	PUSCH Pkt					Present Pkt		
	PRACH Pkt					Mgmt. Pkt		
0x002	Timing Pkt	Rsvd	Rsvd	Rsvd	Rsvd	Ctrl Pkt	Rsvd	Rsvd
	PUSCH Pkt					Present Pkt		
	PRACH Pkt					Mgmt. Pkt		

Valid CPRI Link ID's and hence VLAN TAG ID's will be from 0 to 15. The Packet Type definitions are listed in Table 4.2.

Table 4.2. Packet Type Definitions mapping to PCP field in VLAN Tag

Packet Type	PCP field in VLAN Tag [2:0]
Timing Packet	0x7
PUSCH Packet	0x7
PRACH Packet	0x7
Present Packet	0x2
Control Packet (CSR)	0x2
Management Packet	0x2
LTE Time Domain Packet	0x7

4.2.1.1 Packet Sub-Types

In R1.0.0 some packets are required to be assigned to the same VMDQ, as such they have the same PCP value. In order for SW to distinguish between these packets there will be a 16 bit sub-type field used.

Table 4.3. Packet Sub-Type Definitions

Packet Sub-Type	Sub-Type
PDSCH Packet	0x0000
PDCCH Packet	0x0001
PBCH Packet	0x0002
Timing Packet	0x0003
PSS Packet	0x0004
RS Packet	0x0005
CPRI Data Packet	0x0006
LTE Time Domain Packet	0x0007
Control Specific Register Packet	0x0008
Present Broadcast Packet	0x0009
Management Status Packet	0x000A
PUSCH Packet	0x000B
PRACH Packet	0x000C

4.2.2 Control Packets

4.2.2.1 CSR Access Packet

The CSR Access Packet is used to read and write a register in the RPE device. The SW application sends the CSR Packet Request and the RPE Front end will send a CSR response packet for every CSR request it receives. The CSR response will indicate success of a CSR request and may also include data in the case of a read request. If there is no CSR Response to a CSR Read or Write this indicates a failure of the CSR Request. The fields of the CSR response will be the same as the CSR request. Figure 4-5 shows the CSR packet definition. In this example the VLAN ID is set to 0x001 indicating CPRI link 1 and the VLAN priority set to 0x2 indicating a control packet type as defined in Table 4.2.

Figure 4-5. CSR Packet Definition

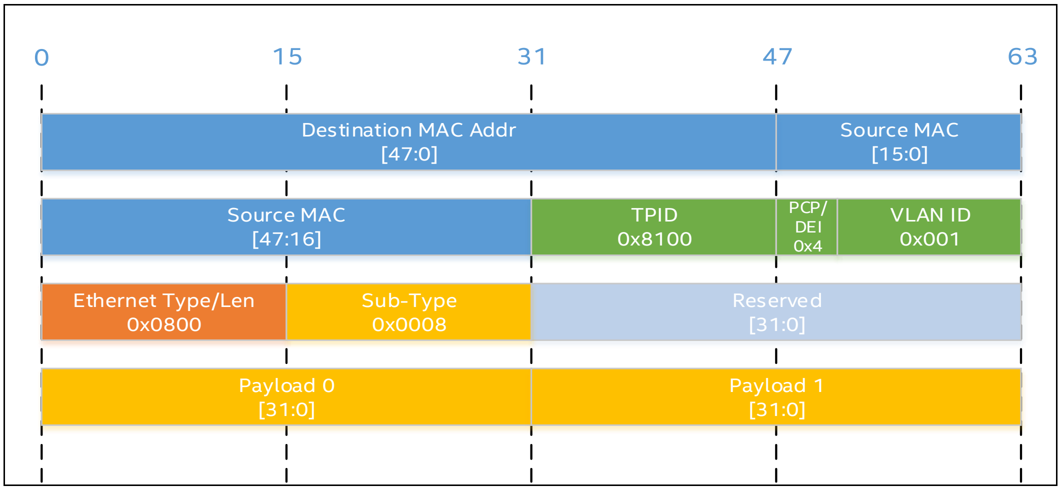


Figure 4-6. CSR Payload 0

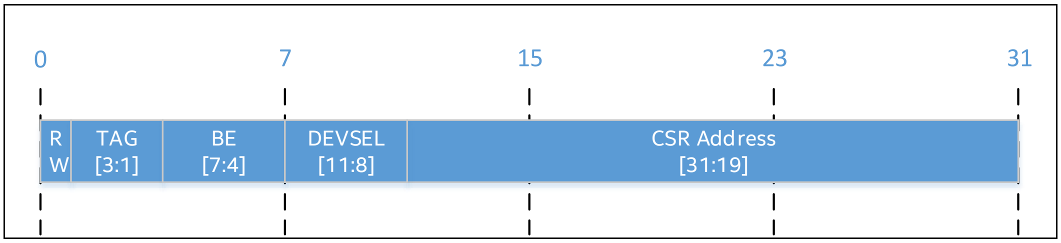


Table 4.4. CSR Payload description

CSR Payload 1 Field	Description
RW	1 bit field to indicate whether the CSR packet is a read request or a write request.

TAG	3 bit field assigned by SW to indicate the sequence of a CSR request. Not touched by HW.
BE	4 bit mask indicating which byte(s) in the register addressed by "CSR Address" that will be written. Currently this feature is not implemented in HW.
DEVSEL	4 bit field indicating the Device selection. The Devices to choose from are listed in The Device Select Options are listed below in Table 4.5, which shows the values used to select a particular device. Each device corresponds to a different component in the Remote Processing Engine, and each device has it's own set of internal registers that can be accessed separately. For example the CPRI device corresponds to the CPRI IP in the RPE, by selecting this device the CSR packet will be directed to the CPRI IP. In this case the offset address used must be a valid CPRI IP offset address. Table 4.5.
CSR Address	19bit address of the selected register.

The Device Select Options are listed below in Table 4.5, which shows the values used to select a particular device. Each device corresponds to a different component in the Remote Processing Engine, and each device has it's own set of internal registers that can be accessed separately. For example the CPRI device corresponds to the CPRI IP in the RPE, by selecting this device the CSR packet will be directed to the CPRI IP. In this case the offset address used must be a valid CPRI IP offset address.

Table 4.5. Device Select Options

Device	4 bit Flag
Ethernet PHY	000
Ethernet MAC	0001
CPRI	0100
RPE Local Registers	1111

The Payload 1, a 32-bit field contains the 32-bit value to be written to the selected register when the request is a write. When the request is a read it should be set to 0x0000. The CSR response packet will follow the same packet definition with the Payload 1 containing the value that was read from the selected register. In the case of a write it will be set to 0x0000.

4.2.2.2 Present Broadcast Packet

The present broadcast packet is sent by the Remote Processing Engine every 1 second onto the network, its purpose is to indicate the presence of the RPE device on the network. Included in the Present Packet is information about the RPE device, its features and capabilities. The present packet is only sent when the RPE is not paired with an application. Pairing occurs when an application writes a register to disable the Present Broadcast packet, this is discussed later.

Figure 4-7. Present Packet

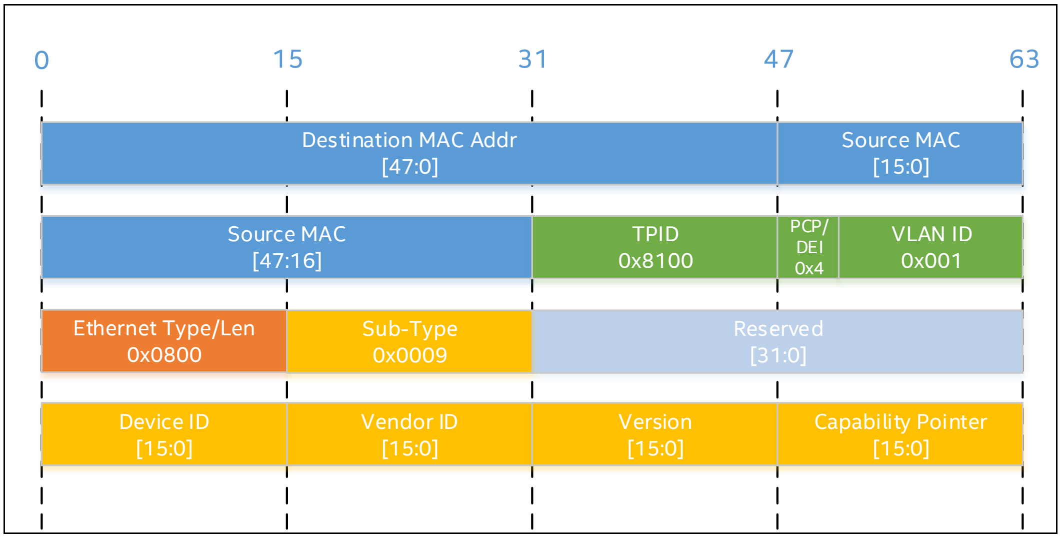


Table 4.6. Present Packet Description

Field	Description
Device ID	16 bit field containing an Identification number for this device.
Vendor ID	Intel 0x8086 vendor ID.
Version	16 bit value indicating the version of the HW.
Capability Pointer	16 bit mask indicating the capabilities of the device. This is defined in the API specification.

The Capability pointer is a 16bit mask that indicates the capabilities of the device. The bit mask is explained in the Table 4.7. If a bit is set to TRUE it indicates the capability or function is available in the device.

Table 4.7. Capability Pointer

Bit	Definition
0	Indicates CPRI Interface is available on this Device
1	LTE PUSCH/PDSCH Interface with resource block selection is available in the device
2	LTE PSS/RS/SSS Interface is available on this Device
3	LTE PRACH Interface is available on this device
4 to 15	Reserved

4.2.3 Data Packets

4.2.3.1 Timing Packets

Timing packets are sent from the RPE device every 1ms to indicate the start of a sub-frame. The 1ms second boundary is aligned to the CPRI Radio Frame pulse. The timing packet will be used by software to trigger the start of the next 1ms worth of data to be transmitted to the RPE. In the current release software performs the packet classification on all data packets, it is for this reason that the timing packets have a priority set to the same as the data packets 0x7. This means the timing packets will be received on the same VMDQ as the data packets, which will not impact the software application.

Figure 4-8. Timing Packet Definition

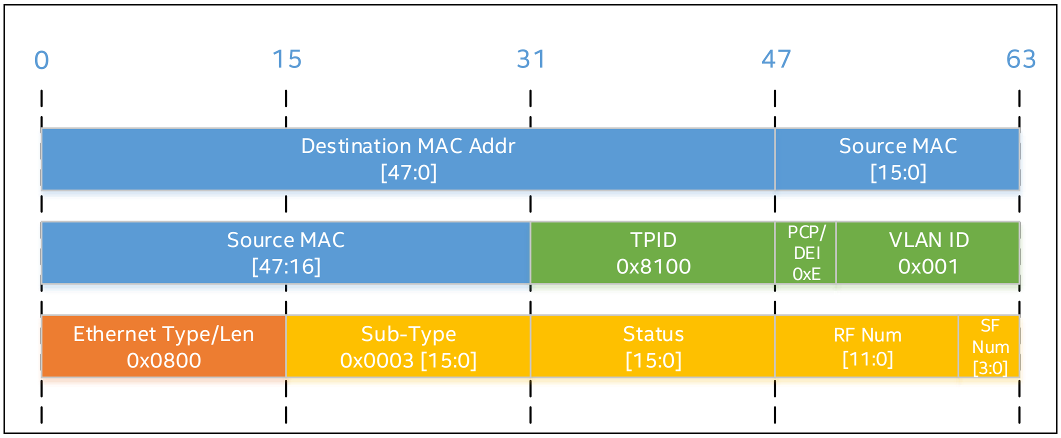


Table 4.8. Timing Packet Field definition

Field	Description
Status	32bit field that contains various status of the RPE device and the IO interfaces. This is explained in Table 4.9,
RF Num	Indicates the current radio frame number.
SF Num	Indicates the current subframe number within the current radio frame.

Table 4.9. Timing Packet Status Fields

Field	Description
BIT [0] Downlink Synchronization	Status of the current synchronization on the CPRI link. ‘1’ means synchronized, ‘0’ means no synchronization

BIT [1] DL FIFO Overflow	Indicates an overflow error condition in the downlink. ‘0’ mean no overflow condition, ‘1’ means that an underflow condition has occurred.
BIT [2] DL FIFO Underflow	Indicates an underflow error condition in the downlink. ‘0’ mean no underflow condition, ‘1’ means that an underflow condition has occurred.
BIT [3] PRACH FLOW ERROR	Indicates the FLOW ERROR condition triggered from the PRACH RTL module inside the timing packet window.
BIT [4] PRACH FRAME ERROR	Indicates the FRAME ERROR condition triggered from the PRACH RTL module inside the timing packet window.
BIT [6:5] First PRACH ERROR Identifier	“00”: No Error identified “01”: FLOW Error is the 1st received in Timing Pkt “10”: FRAME Error is the 1st received in Timing Pkt “11”: Both Errors occur at same time in Timing Pkt
BIT [15:7]	Reserved for future use.

When the FBR is configured up in BYPASS MODE, the following bits are contained in the STATUS field:

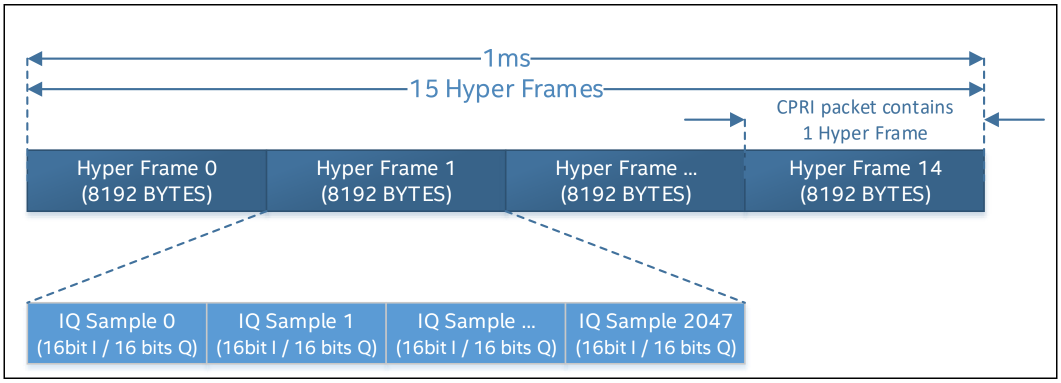
Table 4.10 Timing Packet Status Fields BYPASS MODE

Field	Description
BIT [0] Downlink Synchronization	Status of the current synchronization on the CPRI link. ‘1’ means synchronized, ‘0’ means no synchronization
BIT [1] DL RD ERROR	Indicates a READ ERROR on the down link path
BIT [2] DL WR ERROR	Indicates an WRITE ERROR on the down link path
BIT [3] UL WR ERROR	Indicates a WRITE ERROR on the Uplink path
BIT [4] PRACH FLOW ERROR	Indicates the FLOW ERROR condition triggered from the PRACH RTL module inside the timing packet window.
BIT [5] PRACH FRAME ERROR	Indicates the FRAME ERROR condition triggered from the PRACH RTL module inside the timing packet window.
BIT [7:6] First PRACH ERROR Identifier	“00”: No Error identified “01”: FLOW Error is the 1st received in Timing Pkt “10”: FRAME Error is the 1st received in Timing Pkt “11”: Both Errors occur at same time in Timing Pkt
BIT [15:8]	Reserved for future use.

4.2.3.2 CPRI Data Packets

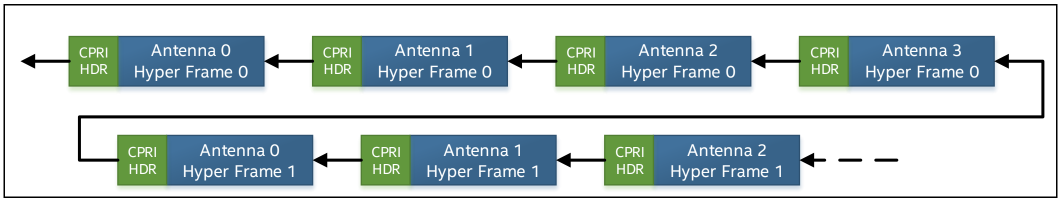
When the RPE is configured with LTE features disabled the RPE allows direct communication from Ethernet to the CPRI interface. In this scenario the RPE expects time-domain IQ Samples arranged in the Ethernet packets aligned with the format expected by the CPRI interface. This is shown in Figure 4-9. Each CPRI Packet contains a single hyper-frame worth of data for a single antenna.

Figure 4-9. CPRI Hyper Frame Structure



All the CPRI packets are expected to arrive in sequence for the first release of the RPE. This means that the SW application needs to ensure the packets are sent in sequence and interleaved per antenna. The sequence requires that antenna data is interleaved, as described in Figure 4-10.

Figure 4-10. CPRI Packet Sequence



The CPRI packet definition follows in Figure 4-11, the same packet definition is used for both upstream and downstream communications. Each packet contains one hyper-frame data for a single antenna carrier.

Note: Due to the size of a hyper-frame it requires that Ethernet Jumbo frames be used when communicating with the RPE. This is true for both CPRI packets and LTE packets.

Figure 4-11. CPRI Packet Definition

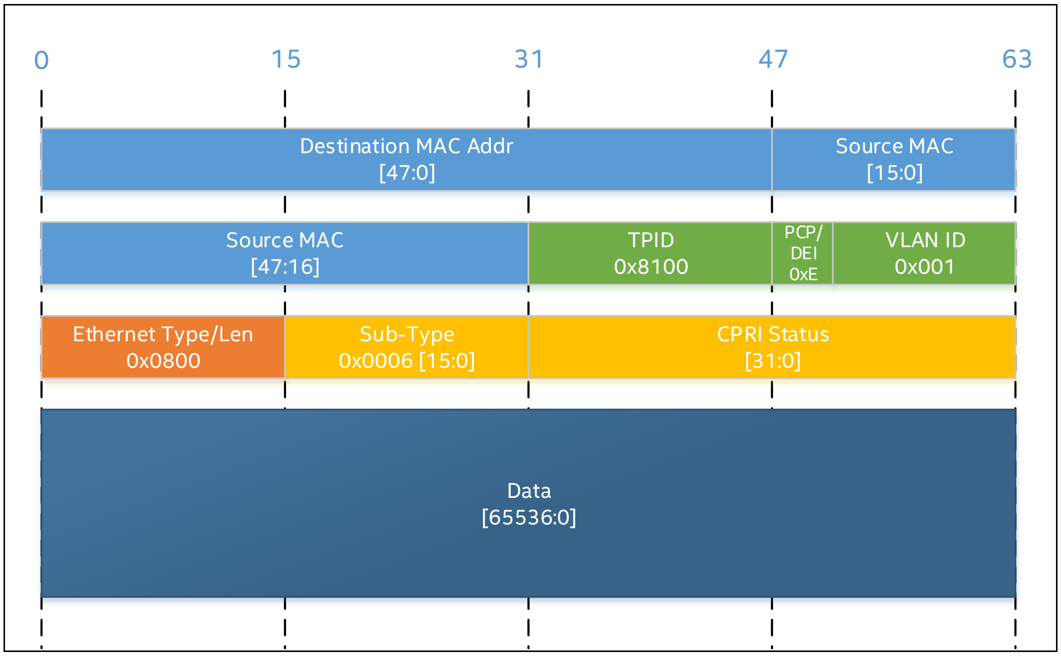


Figure 4-12. CPRI Status Field Description

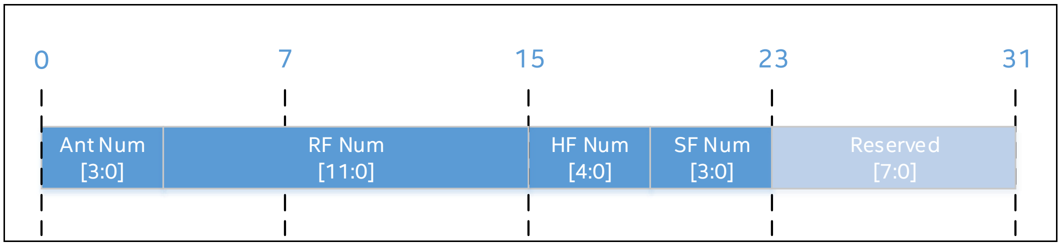


Table 4.11. CPRI Packet Field Description

Field	Description
Sub-Type	16bit value containing the sub-type of packet. The sub-type for CPRI Data packets is defined at 0x0006. This field should not change per TTI.
CPRI Status	32 bit value containing information about the current CPRI packet and the status of the link.
CPRI Status : Ant Num	The Antenna Number that this hyper frame belongs to. Range 0 to 8. For first release only 4 antenna carriers are support.
CPRI Status : RF Num	The Current Radio Frame number that this hyper frame belongs to. Range 0 to 4095

CPRI Status : HF Num	The current Hyper Frame number that the packet corresponds to within the current radio frame. Range 0 to 14.
CPRI Status : SF Num	The current Sub Frame Number that this packet belongs to. Range 0 to 9.
CPRI Status : Status	8 bit field indicating error conditions in on the CPRI Interface. The possible values are outlined in Table 4.12

Table 4.12. CPRI Status Values

Value	Description
0x00	No Errors detected.
0x01	Underflow Error Detected
0x02	Overflow Error Detected
0x03	Loss of Signal Detected

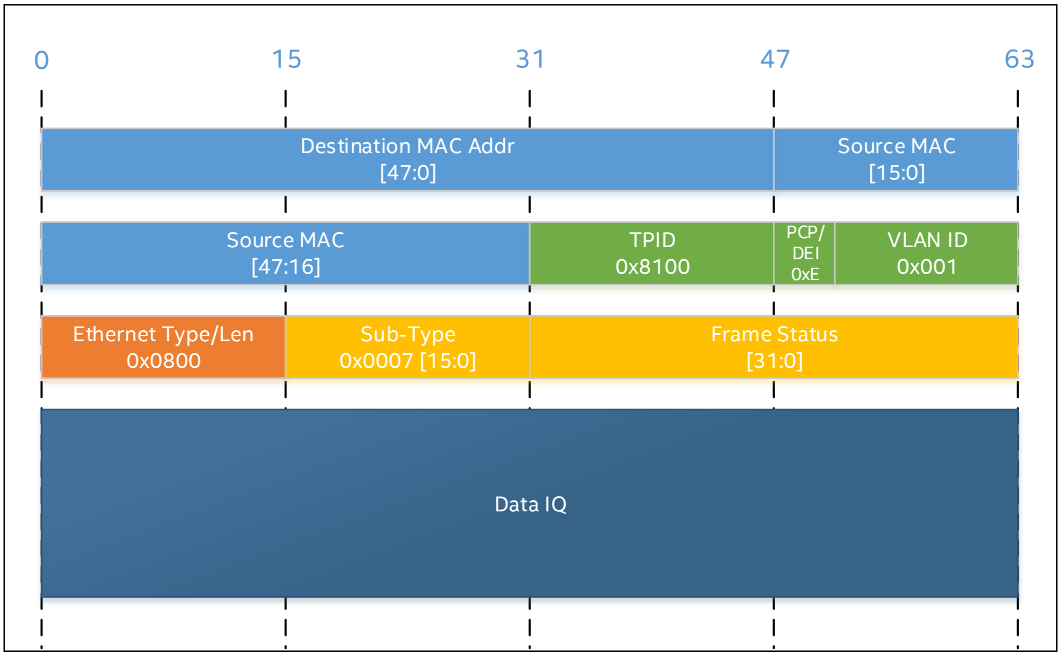
In the event of receiving a Loss of Signal error, it is recommended the SW to reset the Remote Processing Engine. This can be done through register writes.

4.2.3.3 LTE Data Packets

4.2.3.3.1 LTE Time Domain Packet Format

The LTE Time Domain packets will be used when Bypass mode is enabled in the Remote Processing Engine. In this mode the LTE features are bypassed and Ethernet packets are passed directly to the CPRI interface. In this case the CPRI Hyper frame packets are mapped to LTE Symbols and encapsulated in Ethernet headers and sent to the Software Application. Similarly the SW application sends LTE Symbols to the Remote Processing Engine which will in turn map the LTE symbols to CPRI Hyper frames to be sent over the CPRI Interface.

Figure 4-13. LTE Time Domain Packet Format



The Frame status is outlined in more detail in Figure 4-14 and Table 4.13.

Figure 4-14. LTE Frame Status

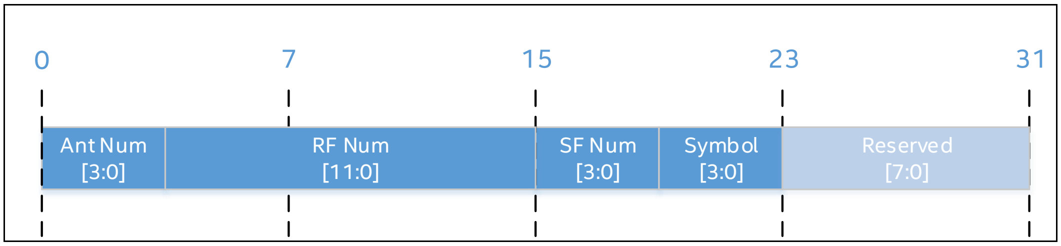


Table 4.13. LTE Frame Status definition

Field	Description
Ant Num	4 bit field indicating the antenna number this particular LTE symbol belongs to. Valid Antenna numbers are 0, 1, 2, 3, 4, 5, 6, 7
RF Num	12 bit field indicating the Radio Frame number this particular LTE symbol belongs to.
SF Num	4 bit field indicating the sub-frame number in the radio frame that this particular LTE Symbol belongs to.
Symbol	4 bit field indicating the LTE Symbol number within the current subframe that this LTE Symbol is. Valid range of 0 to 13.

Reserved	Reserved for future use.

4.2.3.3.2 LTE PUSCH Packet Format

The LTE PUSCH Packet format is outline below. LTE PUSCH packets are sent from the Remote Processing Engine to the IA server. Once the CPRI link is configured (with bypass mode disabled) and started the Remote Processing Engine will send LTE PUSCH packets. Each millisecond several LTE PUSCH packets will be sent, depending on the configuration and the number of Resource Blocks being used. All the IQ samples in this packet will be compressed IQ samples – SW will need to decompress the IQ samples before use. Each PUSCH packet contains all the necessary RB's for a single Symbol, no more than one Symbol per Antenna should be sent per PUSCH packet. Resource Blocks and Antenna data must be contiguous in the packet.

Figure 4-15. LTE PUSCH Packet Format

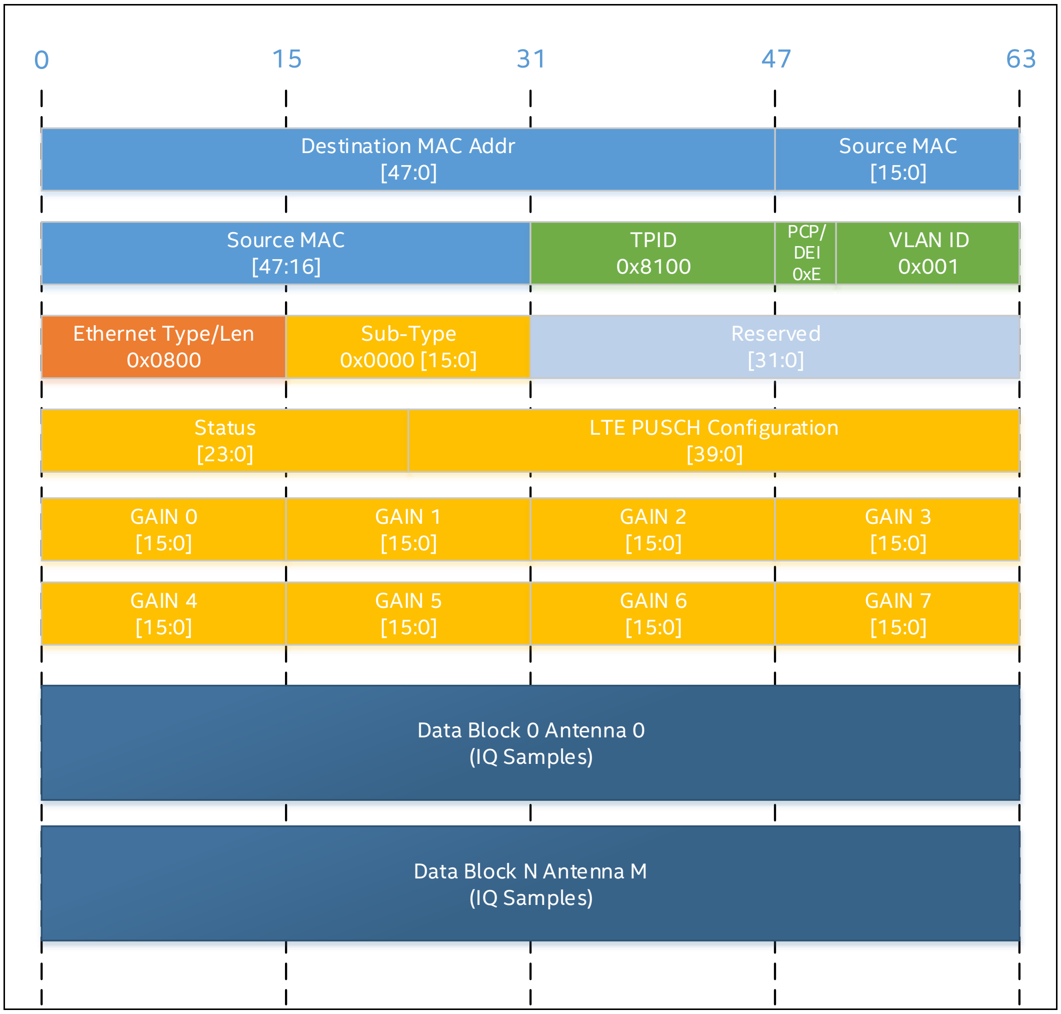


Table 4.14. LTE PUSCH Status Description

Field	Description
Status[0]	Reserved
Status[1]	If TRUE indicates that this Subframe is an “S Frame” For Phase 1 S Frames will not be supported.
Status[23:2]	Reserved

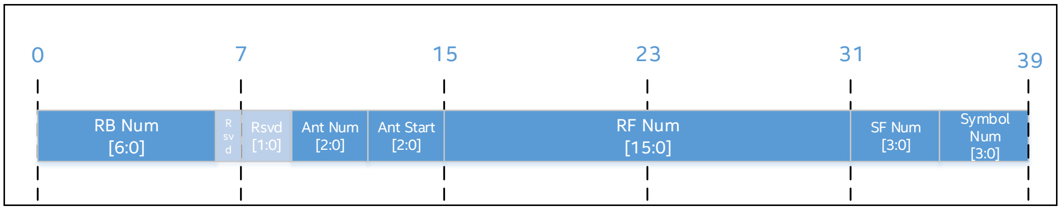


Figure 4-16. LTE PUSCH Configuration Format

Table 4.15. LTE PUSCH Configuration Details

Field	Description
RB Num	The number of resource blocks in the LTE PUSCH Packet.
Rsvd1 [0]	1 Bit Reserved
Rsvd2 [1:0]	2 But Reserved field
Ant Num	The number of Antenna Carriers represented in the packet. Antenna numbers range from 0 to 7 with valid inputs being 1, 3 and 7. Where 1 indicates 2 active antenna, 3 indicates 4 active antenna and 7 indicates 8 active antenna.
Ant Start	The starting antenna number. Example if “Ant Num” equals 2 and “Ant Start” equals 2, it means this PUSCH packet contains the antenna numbers 2 and 3. Antenna data must be in sequence in the packet.
RF Num	Current Radio Frame Number that this LTE packet belongs to.
SF Num	Current Sub-frame number that this LTE packet belongs to.
Symbol Num	Current LTE Symbol number that this LTE packet belongs to.

Table 4.16. LTE PUSCH GAIN

Field	Description
GAIN[8:0]	AGC output
GAIN[9]	Reserved
GAIN[15:10]	The AFFT module exp output

LTE PUSCH data block contains the compressed IQ samples. Each sample consists of 8bit real and 8 bit imaginary data making up the 16bit IQ sample. The pattern is outlined in Figure 4-17.

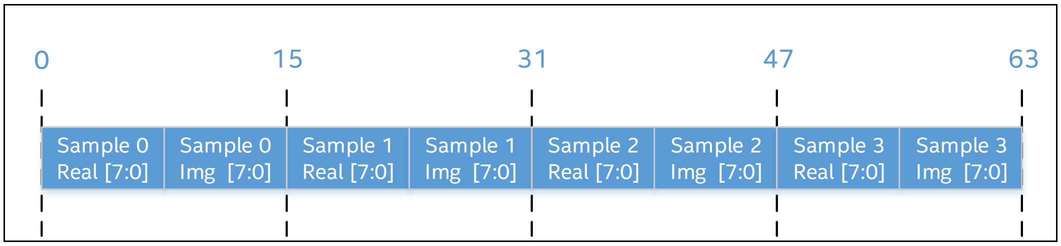


Figure 4-17. PUSCH Data Block Description

4.2.3.3.3 LTE PRACH Packet Format

For Phase 1 PRACH packets will not be supported. It is up to the application to perform all PRACH processing on the allocated resource blocks received from the PUSCH packet.

The LTE PRACH Packet will be sent from the Remote Processing device to the IA server. The LTE configuration used (PRACH Format) will determine the number and frequency of LTE PRACH packets sent from the RPE.

Figure 4-18. LTE PRACH Packet Format

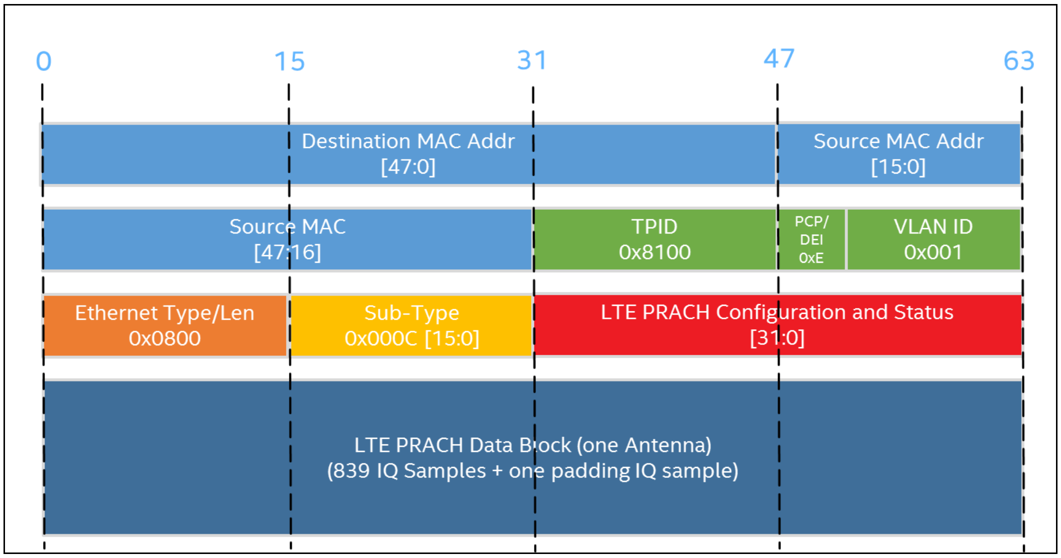

0
15
31
47
63
Destination MAC Addr [47:0]
Source MAC Addr [15:0]
Source MAC [47:16]
TPID 0x8100
PCP/ DEI 0xE
VLAN ID 0x001
Ethernet Type/Len 0x0800
Sub-Type 0x000C [15:0]
LTE PRACH Configuration and Status [31:0]
LTE PRACH Data Block (one Antenna) (839 IQ Samples + one padding IQ sample)

Figure 4-19. LTE PRACH Configuration

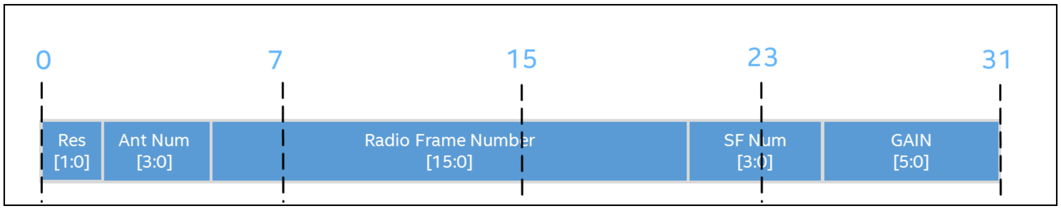


Table 4.17. LTE PRACH Configuration Details

Field	Description
reserved	2 bit field reserved for future use.
Radio Frame Number	16 bit value indicating the current Radio Frame number this antenna data belongs to.
SF Num	4 bit number indicating the sub-frame within the current radio frame that this antenna data belongs to.
Ant Num	4 bit number indicating the antenna number that this data corresponds to.

Gain	The gain is to be treated as an exponent to the result. The Value indicates how many left/right shifts (MUL/DIV by 2) were carried out in the FFT chain to optimally position the data within dynamic range of the process pipe. The exponent is a signed number. So a negative number indicates the number of right shifts implemented within the FFT (to prevent saturation) and vice versa. Thus to restore the output sample to the required dynamic range, a compensation equivalent to the opposite operation is to be carried out. For instance, when reported exponent is negative then each and every data element from the corresponding data block has to be shifted left by the absolute value of the exponent and vice versa.

4.2.3.4 LTE PDSCH Packet Format

The PDSCH packet is the main data packet that is sent from the IA server to the Remote Processing Engine. Several PDSCH packets are sent every 1ms to the RPE device, the size and number of packets that are sent every 1ms depends on the configuration and also the number of Resource Blocks that are active within the sub-frame interval.

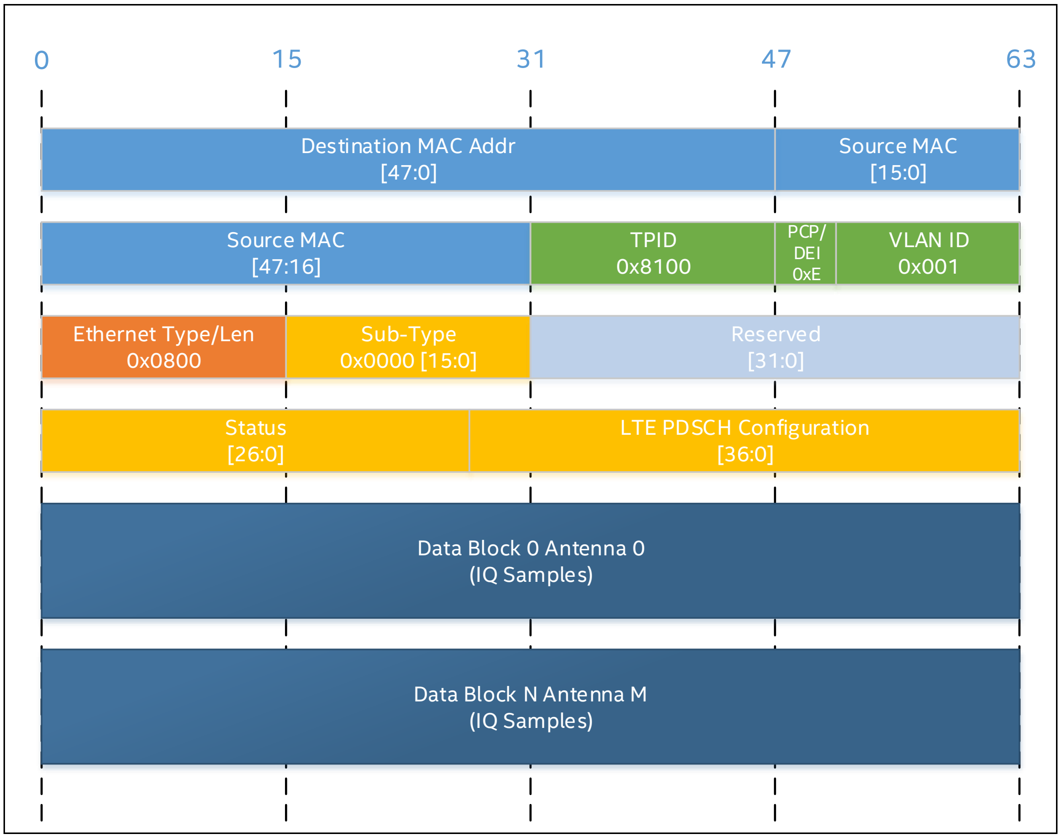


Figure 4-20. LTE PDSCH Packet Format

Table 4.18. LTE PDSCH Status Details

Field	Description
Status[0]	If TRUE enables the RB Selection table
Status[1]	If TRUE Indicates that this is an “S Frame”
Status[26:2]	Reserved

The LTE PDSCH configuration is shown below. This configuration is updated for each packet.

Figure 4-21. LTE PDSCH Configuration

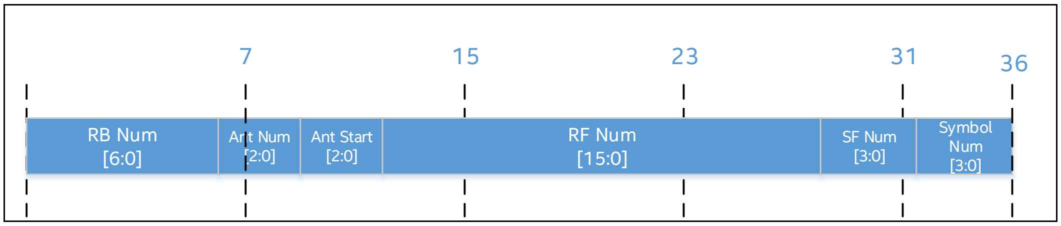


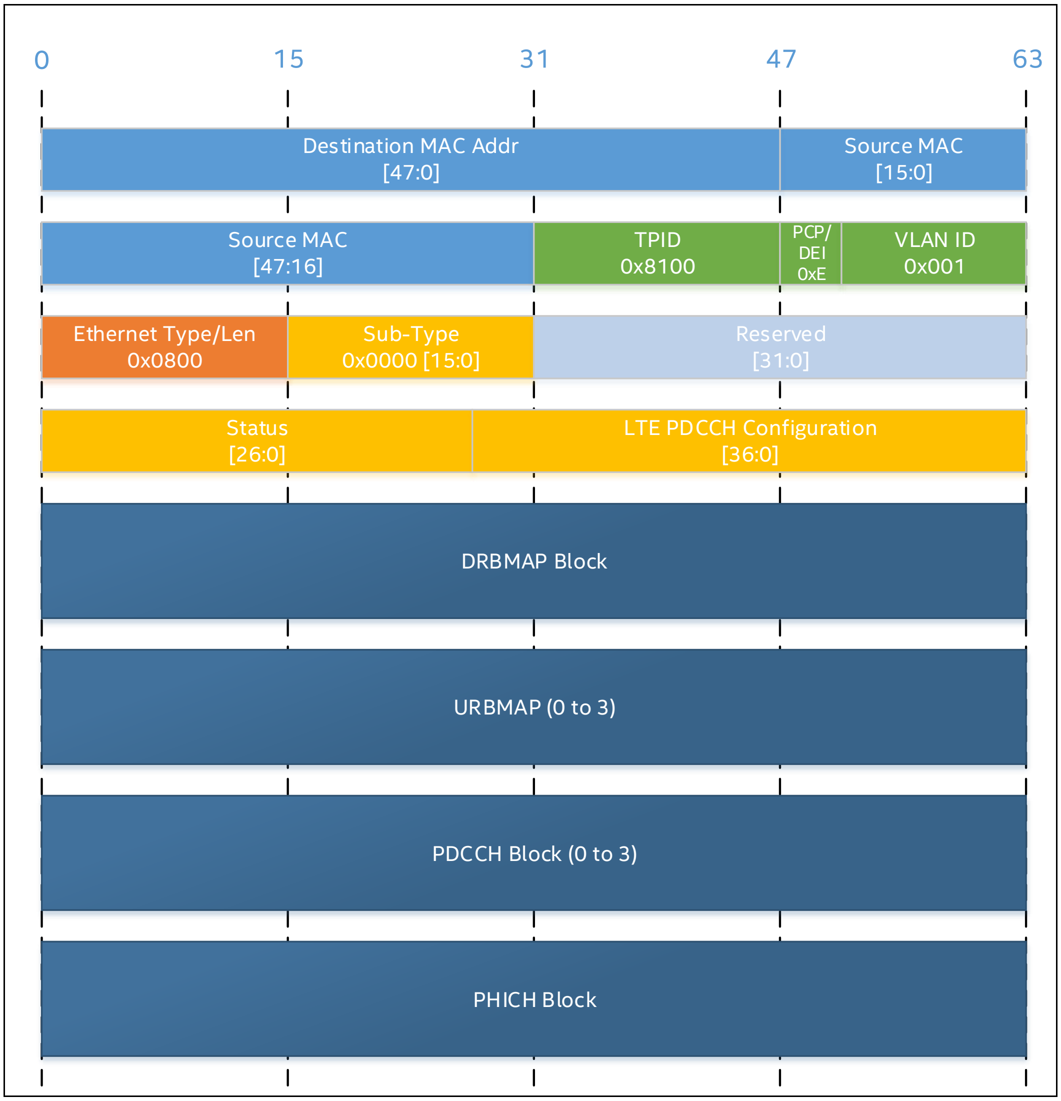
Table 4.19. LTE PDSCH Configuration Description

Field	Description
RB Num	7 bit field indicating the number of Resource blocks present in this PDSCH Packet. The Resource Block length is the same for all RB's. All the RB's for a single LTE Symbol should be included in the packet. RB's must be in sequence.
Ant Num	3 bit filed indicating the number of antennas represented in this PDSCH packet. Not that 0 indicates 1 antenna, 1 indicates 2 antenna and so on.
Ant Start	3 bit field indicating the starting antenna number in this packet. For example is the start antenna number is 2 and the "Ant Num" field is 2 this means the PDSCH packet contains antenna carriers 2 and 3.
RF Num	16 bit number indicating the Radio Frame number this PDSCH packet belongs to.
SF Num	4 bit number indicating the current Subframe number within the radio frame this PDSCH packet belongs to.
Symbol Num	4 bit number indicating the current LTE Symbol that this LTE PDSCH packet belongs to.

4.2.3.5 LTE PDCCH Packet Format

The PDCCH packet is the control packet for downlink, however this packet is also used to send Uplink and Downlink Resource Block mapping information to the LTE Processing in the Remote Processing Engine. It is also used to send PHICH information when it is required. Each PDCCH packet will contain at one Downlink Resource Block Map, between 0 and 3 Uplink Resource Block Maps, between 0 and 3 Downlink Control Blocks and may or may not have PHICH block.

Figure 4-22. LTE PDCCH Packet format



The PDCCH status and configuration fields are described below.

Figure 4-23. LTE PDCCH Packet PDCCH Configuration field

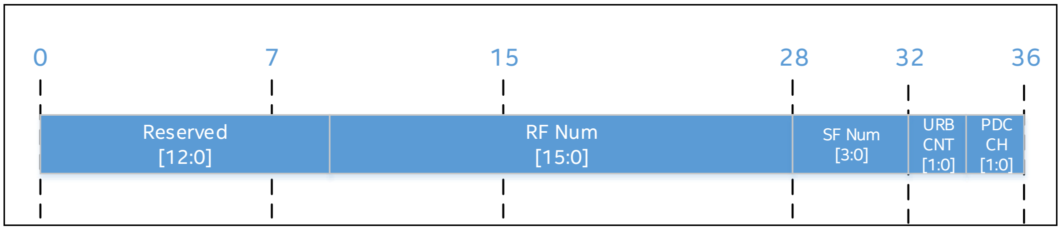


Table 4.20. LTE PDCCH Status Field

Field	Description
Status[0]	Reserved
Status[1]	When set to TRUE indicates that this sub-frame is an S frame
Status[2]	When set to TRUE indicates that this sub-frame includes PBCH Channel
Status[3]	When set to TRUE indicates that this sub-frame includes PSS Channel
Status[26:4]	Reserved

Table 4.21. LTE PDCCH configuration Field

Field	Description
Reserved	12 bit reserved field.
RF Num	16-bit radio frame number that this packet belongs to.
SF Num	4 bit field indicating the current Sub-frame number within the current radio frame that this packet belongs to.
URB CNT	2 bit field indicating the number of URBMAP blocks that are present in this packet. Values from 0 to 3.
PDCCH CNT	2 bit field indicating the number of PDDCH blocks that are present in this packet. Values from 0 to 3.

Figure 4-24 shows the DRB MAP block format that is used. The format uses 2 bits to represent each Resource Block within the Resource Block Map. The value of the 2 bits indicates whether the Resource block is in use and if it is what table to use to decompress the resource block.

Table 4.22. DRB Block decoding information

RB Value	Description
00	This RB is not used.
01	RB is used
10	Reserved.
11	Reserved

Figure 4-24. LTE PDCCH Packet DRB Map Block Format

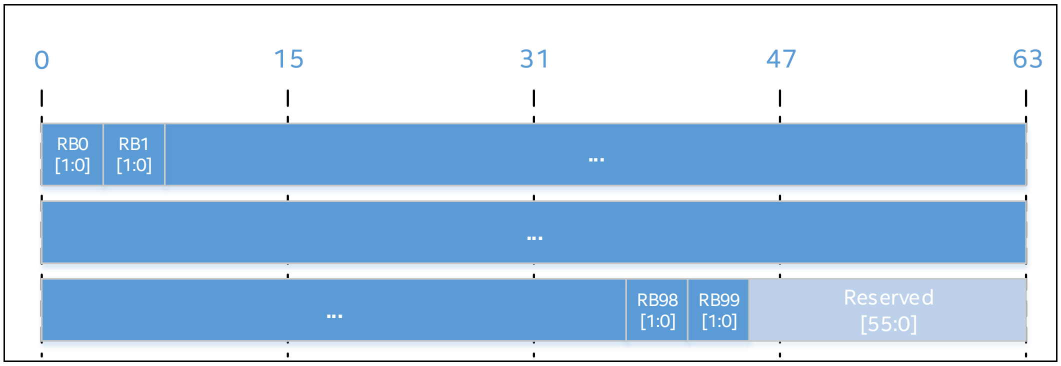
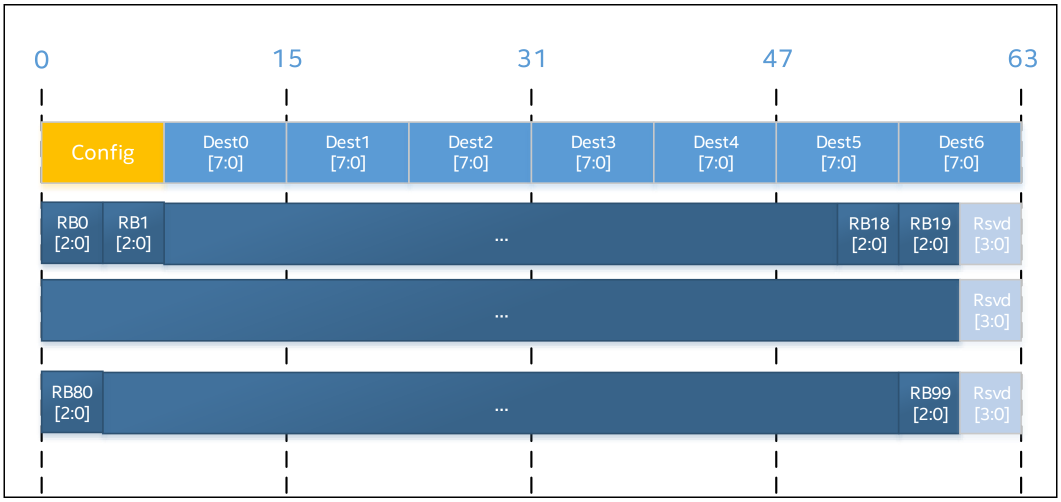


Figure 4-25.. LTE PDCCH Packet URB Map Block



The Uplink Resource Block Map is shown in Figure 4-25. The Resource Block map uses 3 bit fields to indicate the Resource Block usage, whether the resource block is in use and if so what destination server to send the resource block to. A Value of 0 indicates that the resource block is not in used, while a value of 1 to 7 indicates the corresponding destination server to send the resource block to from the selection of Dest0 to Dest6. For the current Phase 1 release only a single destination server is supported Dest0.

Table 4.23. LTE PDCCH Uplink Resource Block PRACH Config Information

Field	Description
Config[0]	Reserved.
Config[1]	"S Subframe" if set to TRUE indicates that the sub-frame indexed by this packet is an S frame
Config[2]	"Nextframe" if set to TRUE this means the URB Block indexes a sub-frame in the next 10ms frame from this PDCCH packet.
Config[6:3]	The sub-frame number indexed by this URB Block
Config[7]	Reserved.

Figure 4-26 and Figure 4-27 show the PDCCH block within the PDCCH packet. The PDCCH block consists of PRB information and PRB mapping information. The PRB processing is divided into 12 xN hints that indicate the processing of a PRB in a PDCCH symbol. The PRB hint is described in

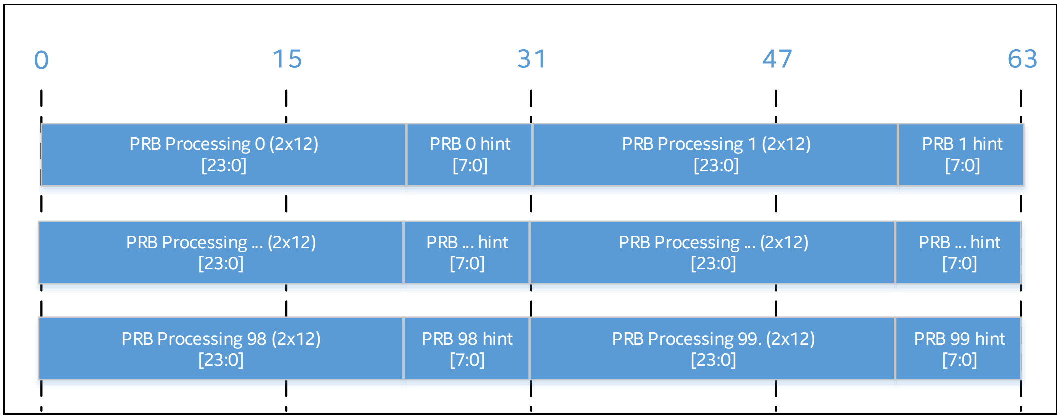

0
15
31
47
63
PRB Processing 0 (2x12) [23:0]
PRB 0 hint [7:0]
PRB Processing 1 (2x12) [23:0]
PRB 1 hint [7:0]
PRB Processing ... (2x12) [23:0]
PRB ... hint [7:0]
PRB Processing ... (2x12) [23:0]
PRB ... hint [7:0]
PRB Processing 98 (2x12) [23:0]
PRB 98 hint [7:0]
PRB Processing 99. (2x12) [23:0]
PRB 99 hint [7:0]

Figure 4-26. LTE PDCCH Packet PDCCH Block

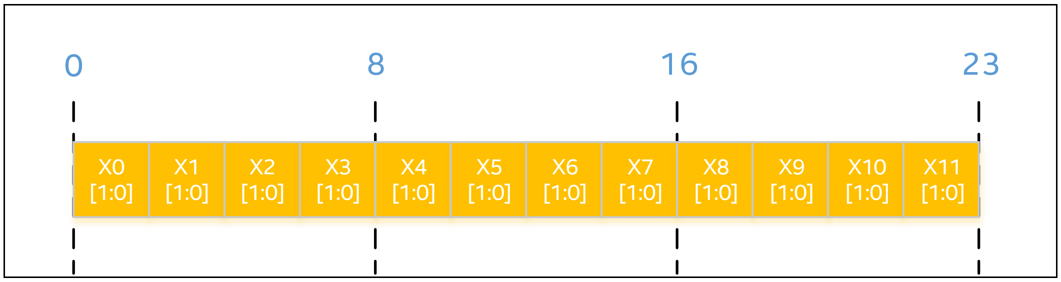


Figure 4-27. LTE PDCCH Packet PDDCH Block PRB processing

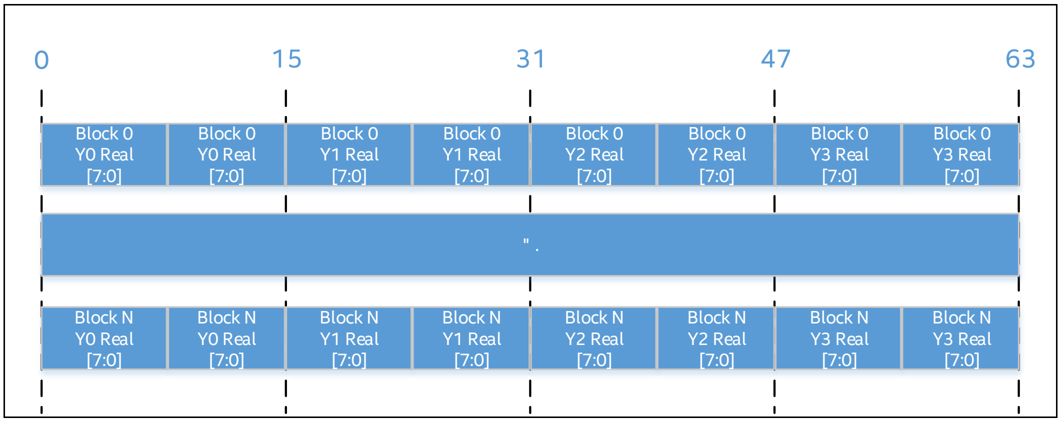
Table 4.24. PRB Hint Detail description

Field	Value	Description
PRB N Hint[7:6]	Symbol-quadruplet grouping method of this PRB	
	00	x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11 is used as the RE data. There are three symbol-quadruplet {x0,x1,x2,x3}, {x4,x5,x6,x7}, {x8,x9,x10,x11}. The RE mapping result is { map(x0), map(x1), map(x2), map(x3), map(x4), map(x5), map(x6), map(x7), map(x8), map(x9), map(x10), map(x11)}
	01	x0,x1,x2,x3, x4,x5,x6,x7,xx,xx,xx,xx is used as the RE data. There are two symbol-quadruplet {x0,x1,x2,x3}, {x4,x5,x6,x7}. The RE mapping result is {0, map(x0), map (x1), 0, map(x2), map(x3)} and {0, map(x4), map(x5),0, map(x6), map(x7)}
	10	x0,x1,x2,x3, x4,x5,x6,x7,xx,xx,xx,xx is used as the RE data. There are two symbol-quadruplet {x0,x1,x2,x3}, {x4,x5,x6,x7}. The RE mapping result is { map(x0), 0, map (x1), map(x2), 0,map(x3)} and {map(x4), 0,map(x5), map(x6), 0,map(x7)}
	11	x0,x1,x2,x3, x4,x5,x6,x7,xx,xx,xx,xx is used as the RE data. There are two symbol-quadruplet {x0,x1,x2,x3}, {x4,x5,x6,x7}. The RE mapping result is { map(x0), map (x1), 0, map(x2), map(x3), 0} and {map(x4), map(x5),0, map(x6), map(x7) ,0}If there is only one PDCCH
PRB N Hint[5:4]	Processing of first symbol-quadruplet	
	00	Replace with zero
	01	QPSK Modulation of this quadruplet
	10	Decompress PHICH x0, x1, x2, x3 and move PHICH read pointer to next line
	11	Decompress PHICH x0, x1, x2, x3 and move PHICH read pointer to next line
PRB N Hint[3:2]	Processing of second symbol-quadruplet	
	00	Replace with zero
	01	QPSK Modulation of this quadruplet
	10	Decompress PHICH x0, x1, x2, x3 and move PHICH read pointer to next line
	11	Decompress PHICH x0, x1, x2, x3 and move PHICH read pointer to next line
PRB N Hint[1:0]	Processing of third symbol-quadruplet if PRBx Hint[7:6]=00	
	00	Replace with zero
	01	QPSK Modulation of this quadruplet
	10	Decompress PHICH y0, y1, y2, y3 and move PHICH read pointer to next line
	11	Decompress PHICH y0, y1, y2, y3 and move PHICH read pointer to next line

Figure 4-28 shows the PHICH Block within the PDCCH packet. Each PDCCH packet has one PHICH block the length of which varies from 0 to 75 Quad words. When the Hint in PDCCH indicates that

PHICH is used (Table 4.24. PRB Hint Detail description) for a PRB this block is then decompressed and used.

Figure 4-28. LTE PDCCH Packet PHICH Block



4.2.3.6 LTE PBCH Packet

The PBCH Downlink broadcast channel is shown in Figure 4-29. This packet is used to update the Remote Processing Engine internal PBCH RAM as such is has no strict timing requirements. Software must ensure the PBCH RAM content is valid before it is used by the LTE interface in the RPE device. Software can sent this packet to the RPE before the CPRI interface is initialized and/or synchronized. The PBCH RAM content will be used to generate the PBCH symbols and insert into the center 72 samples of sub frame 0 on OFDM symbols 7,8,9 and 10. The PBCH is enabled via an API call where the QPSK vector is initialized. The value of the 2bit PBCH RE[0:1] in the PBCH packet will determine the value of the output sample. See 4.2.4 for more information on the QPSK encoding.

The PBCH data block in the packet contains 72 2-bit RE samples that are used to generate the 72 samples inserted into the resource block map.

Note: Depending on the RS Offset selected in 4.2.3.8.1 symbol 8 in the PBCH block will have some samples set to zero as per the LTE spec. There is an internal register used to program this offset that will be abstracted from SW.

Figure 4-29. LTE PBCH Packet Definition

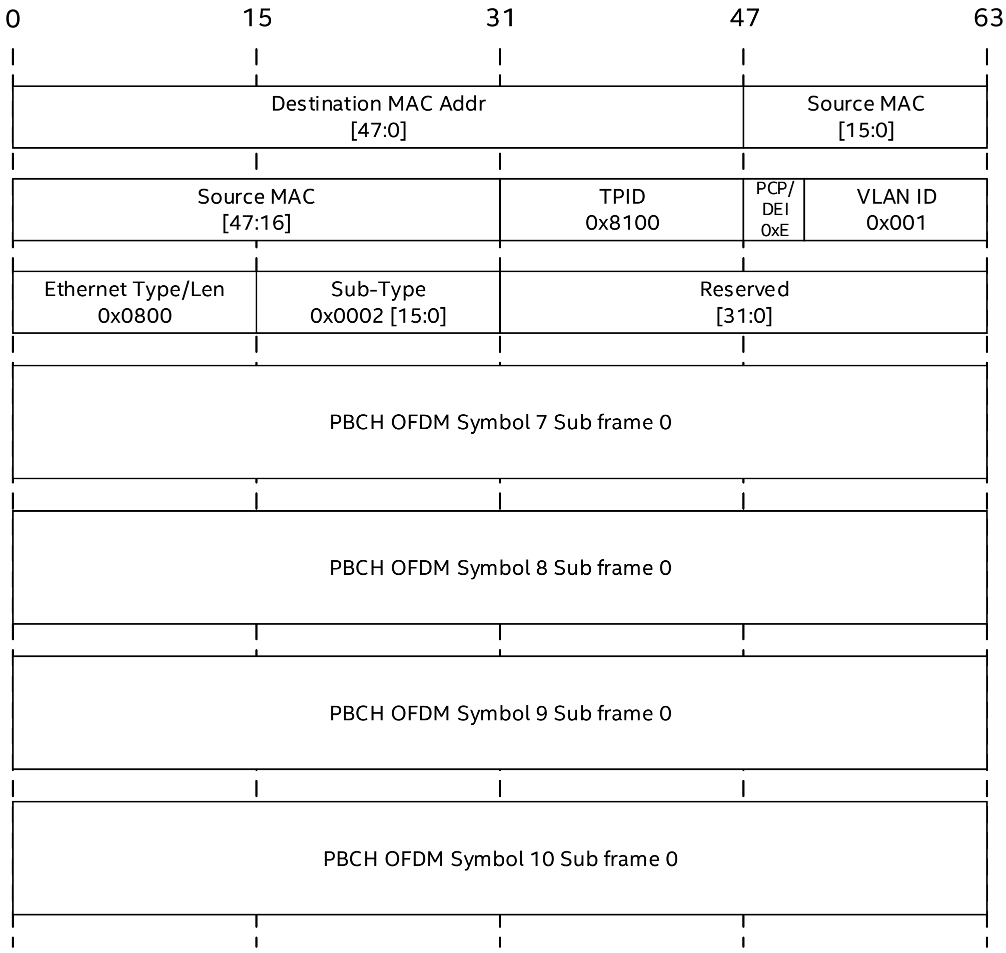
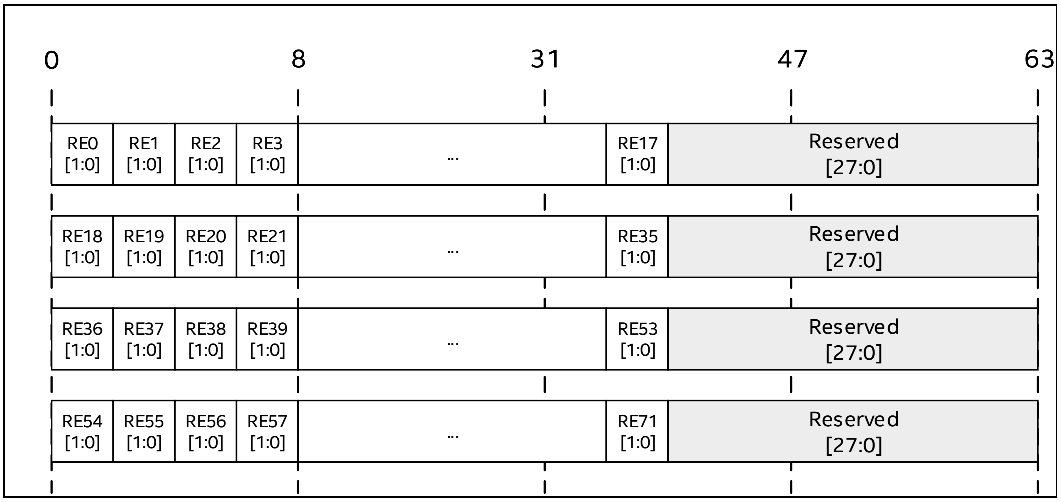


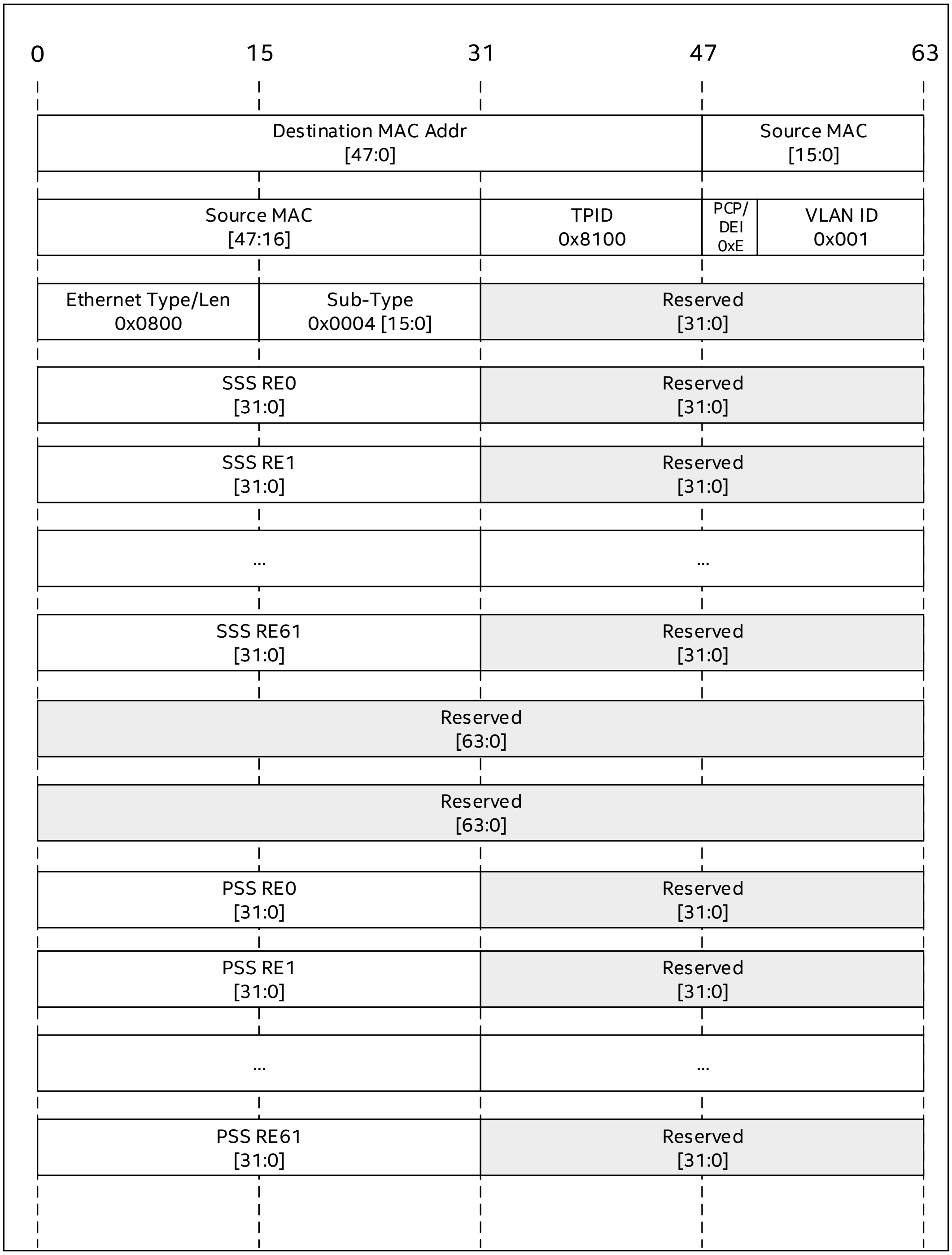
Figure 4-30. LTE PBCH Symbol Format



4.2.3.7 LTE PSS Packet

The downlink Synchronization packet PSS is listed below in Figure 4-31. The PSS packet is used to update the PSS RAM and SSS RAM in the LTE interface on the RPE device. As such it has no strict timing requirements. When the LTE PSS channel is required in a sub frame the RPE device will use the RAM content to fill the corresponding symbol. Software must ensure the RAM content is valid before it is used. Software can fill the RAM buffer before the CPRI link is configured/synchronized. The PSS packet contains both the Primary Sync Signal and the Secondary Sync Signal. The RAM buffer will be used to replace the center 62 samples with the outer samples 0'd making up the center 72 samples for the PSS and SSS channels. The PSS and SSS samples will only be inserted into OFDM symbols 5 (SSS) and 6 (PSS) of sub frame's 0 and 5 within one radio frame.

Figure 4-31. LTE PSS Packet Definition

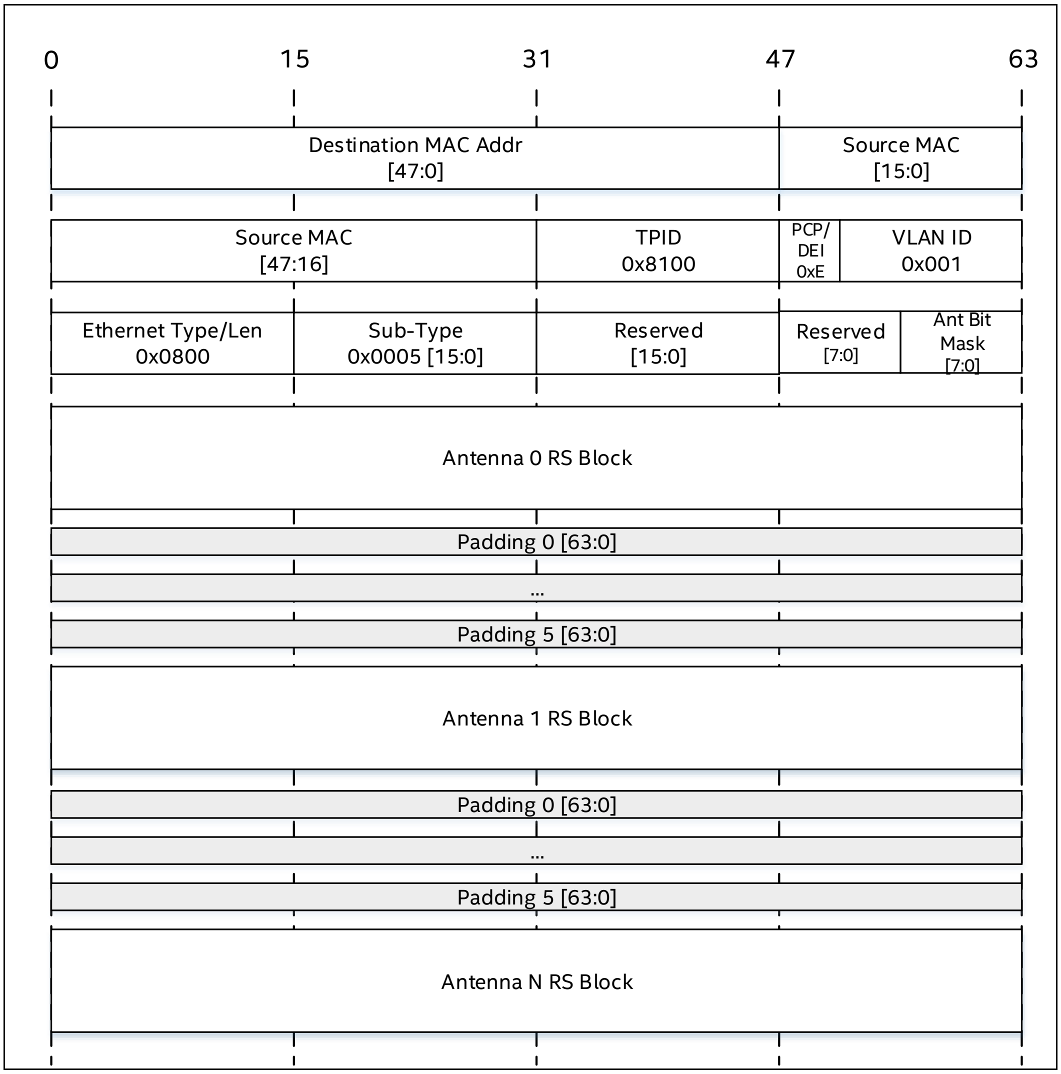


<table>
<tr><th>0</th><th>15</th><th>31</th><th>47</th><th>63</th></tr>
<tr><td colspan="3">Destination MAC Addr
[47:0]</td><td colspan="2">Source MAC
[15:0]</td></tr>
<tr><td colspan="2">Source MAC
[47:16]</td><td>TPID
0x8100</td><td>PCP/
DEI
0xE</td><td>VLAN ID
0x001</td></tr>
<tr><td>Ethernet Type/Len
0x0800</td><td>Sub-Type
0x0004 [15:0]</td><td colspan="3">Reserved
[31:0]</td></tr>
<tr><td colspan="2">SSS RE0
[31:0]</td><td colspan="3">Reserved
[31:0]</td></tr>
<tr><td colspan="2">SSS RE1
[31:0]</td><td colspan="3">Reserved
[31:0]</td></tr>
<tr><td colspan="2">...</td><td colspan="3">...</td></tr>
<tr><td colspan="2">SSS RE61
[31:0]</td><td colspan="3">Reserved
[31:0]</td></tr>
<tr><td colspan="5">Reserved
[63:0]</td></tr>
<tr><td colspan="5">Reserved
[63:0]</td></tr>
<tr><td colspan="2">PSS RE0
[31:0]</td><td colspan="3">Reserved
[31:0]</td></tr>
<tr><td colspan="2">PSS RE1
[31:0]</td><td colspan="3">Reserved
[31:0]</td></tr>
<tr><td colspan="2">...</td><td colspan="3">...</td></tr>
<tr><td colspan="2">PSS RE61
[31:0]</td><td colspan="3">Reserved
[31:0]</td></tr>
</table>

4.2.3.8 LTE RS Packet Definition

The Reference Signal Packet is defined in Figure 4-32 and Figure 4-33. The RS channel is the downlink synchronization channel and this packet is used to fill the RS RAM in the LTE interface in the RPE. Software must ensure the RAM content is valid before it is used. Software can fill the RAM buffer before the CPRI link is configured/synchronized. The RAM Buffer is large enough to store the RS symbols for 1 LTE Frame (10 Sub-frames). As such it is expected that the Antenna RS Block within the RS packet contains 8000 RS symbols, 800 per sub-frame. Padding is added to keep each antenna RS block aligned to 2048 boundary. RS is enabled per antenna carrier.

Figure 4-32. LTE RS Packet Definition

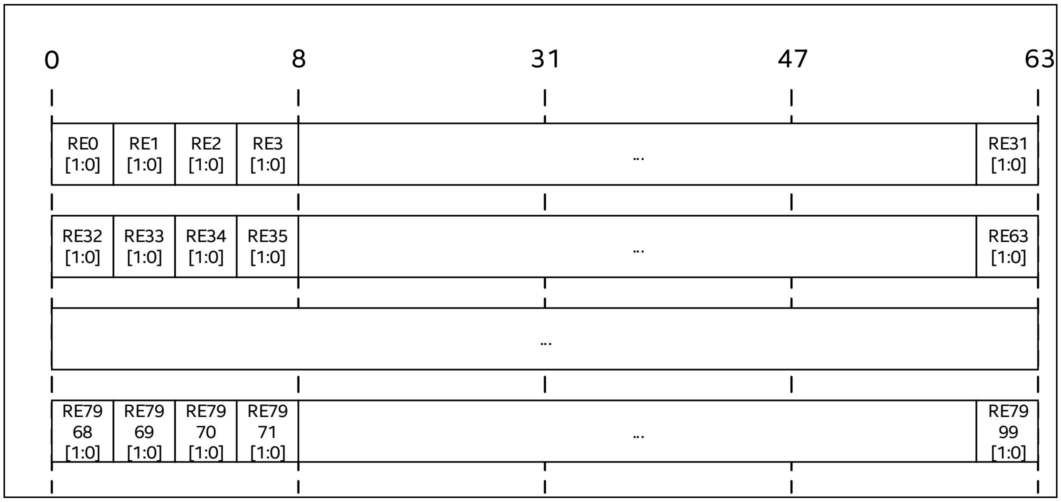


The Ant Bit Mask field is an 8 bit bit-mask that indicates what antenna numbers the RS Packet contains. For the first release only 2 antennas are supported.

Table 4.25. RS Packet Antenna Bit Mask

Bit Mask Value	Active Antenna Numbers
0x01	Antenna Number 0 is the only active antenna carrier
0x03	Antenna Numbers 0 and 1 are the only active antenna carriers.
0x0F	Antenna Numbers 0, 1, 2, and 3 are active antenna carriers.
0xFF	Antenna Numbers 0, 1, 2, 3, 4, 5, 6, and 7 are active antenna carriers.

Figure 4-33. LTE RS Packet RS Block Definition



4.2.3.8.1 RS Offset

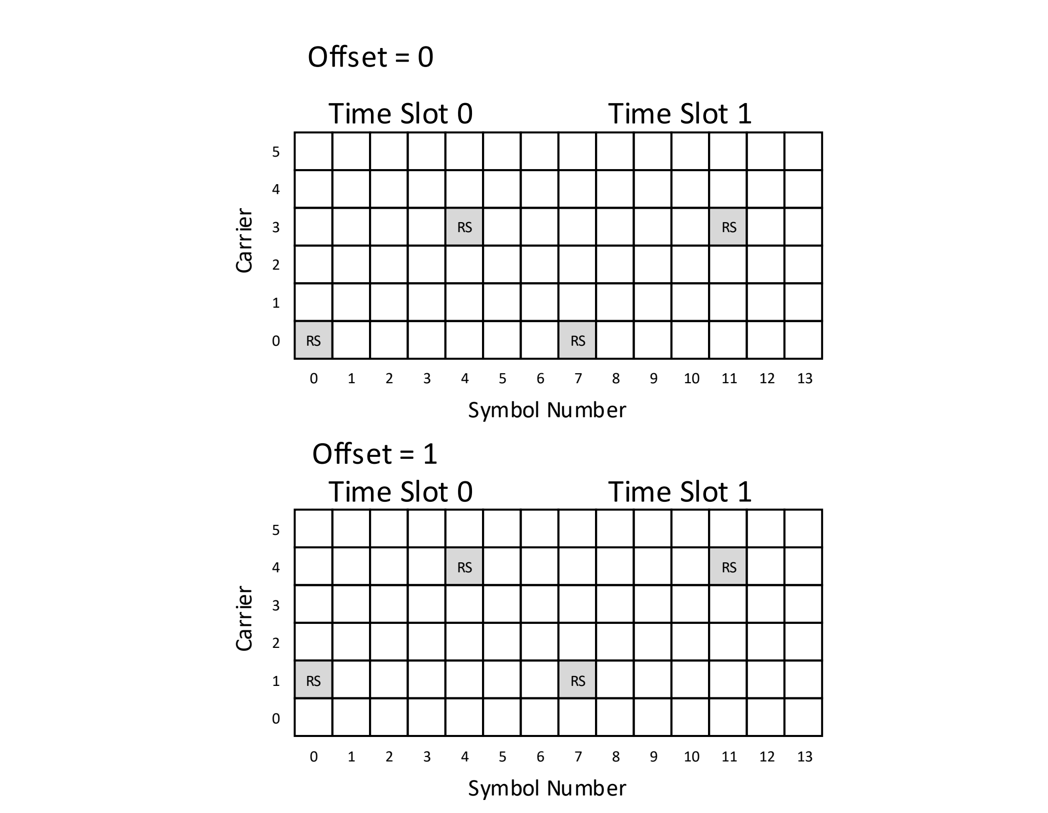
The RS Offset is set at initialization time via an API call

```
RpeStatus rpeLteRsFeaturesSetup(RpeInstanceHandle handle,
RpeLteRsFeaturesConfiguration config);
```

Using the RpeLteRsOffsets offsetConfiguration parameter.

The offset changes the starting position of the RS symbol in the Resource Block Map as per figure below. Note this is for antenna 0 for a 2 antenna system.

Figure 4-34. RS Offset



4.2.3.8.2 RS QPSK

The RS QPSK vector is set at initialization time via an API call

```
RpeStatus rpeLteRsFeaturesSetup(RpeInstanceHandle handle,
RpeLteRsFeaturesConfiguration config);
```

Using the RpeLteRsQpskVector parameter.

4.2.4 QPSK Encoding Algorithm

The following QPSK encoding is performed for the RS, PBCH packets using the associated "QPSK Vector" register that is set through appropriate API.

QPSK Vector Register – individual register for RS, PBCH

QPSK Vector Low [0:15]	QPSK Vector High [15:31]

The value of the 2 bit field *RE[0:1]* in the RS, PBCH packet will determine the 32bit complex output for the RS signal that is input to the Resource Block Map.

Table 4.26. QPSK Encoding Algorithm

RE[0:1]	OUTPUT Low16 [0:15]	OUTPUT High [16:31]
00	qpskVectorLow [0:15]	qpskVectorLow [0:15]
01	qpskVectorHigh [16:31]	qpskVectorLow [0:15]
10	qpskVectorHigh [16:31]	qpskVectorLow [0:15]
11	qpskVectorHigh [16:31]	qpskVectorHigh [16:31]

5 HW SW Interactions

This section is intended to outline the different interactions between HW and SW and in particular capture the expected processing that will occur in the HW, based on the SW inputs. In this section it assumed that bypass mode is not used and all LTE features are enabled.

5.1 Downlink Processing

The downlink processing pipeline was shown in Figure 3-4. The processing is based on LTE Symbol basis. The Software must ensure to send LTE Symbol data for each antenna interleaved, which is in a sequence like Antenna 0, Symbol 0, Antenna 1, Symbol 0 etc. The LTE Symbol data is compressed by the Software before sending to the RPE device. Downlink processing module works at a 1ms timing interval controlled by the 1st PDCCH packet of the corresponding sub-frame. The PDCCH packet contains all the attributes for a particular sub-frame. It is expected SW sends the PDCCH packet enough time before the 1ms sub-frame time, that is the PDCCH packet for sub-frame N+1 should be sent during the processing time of sub-frame N. The buffer to store the PDSCH packet should be large enough to accommodate this "advanced" sending of sub-frame data. In current design the PDSCH buffer used is 64bits * 49152 data buffers which is about 21 symbols of downlink data or ~1.5ms data in time domain. As a BW optimization feature only the RB's that are used are sent as part of the PDSCH packet.

A simple processing flow is listed below:

1. Use either PDCCH or PDSCH packet buffer data to composed symbol data at corresponding CPRI time. If data not available composed an all zero symbol.

2. Replace some RE with PBCH RAM content if necessary

3. Replace some RE with PSS/SSS RAM content if necessary

4. Replace some RE with RS RAM content if necessary

5. Performing IFFT

6. Buffer symbol to different antenna in different buffer and add CP according to sub-frame number. Only Normal Cyclic Prefix is currently supported.

7. Mux out data from each antenna buffer to CPRI Wrapper interface.

5.1.1 De-Compression

De-compression is performed on the received Downlink Data packets from SW (SW is expected to compress the IQ samples before they are sent to the RPE). The decompression is to convert a 8-bit linear PCM value to 16-bit A-law value, the real part and imaginary part will be compressed in parallel. The algorithm of compression is shown as below. Three decompression method will be provide to different user groups. FE shall use DRB table to select one decompression method.

The algorithm of decompression is listed in to function below.

Code Snippet 2. De-compression Algorithm

```
function signed [15:0] alaw2linear;
input [7:0] a_val;
reg [7:0] a_val_1;
reg  [3:0]  quant;
reg [2:0]   seg;
reg  sig;
reg signed [15:0] tmp;
begin
quant = a_val[3:0];
seg  = a_val[6:4];
sig  = ~a_val[7];
case(seg)
 3'h0 : tmp = {8'h0, quant, 4'h0};
 3'h1 : tmp = {8'h1, quant, 4'h0};
 3'h2 : tmp = {7'h1, quant, 5'h0};
 3'h3 : tmp = {6'h1, quant, 6'h0};
 3'h4 : tmp = {5'h1, quant, 7'h0};
 3'h5 : tmp = {4'h1, quant, 8'h0};
 3'h6 : tmp = {3'h1, quant, 9'h0};
 3'h7 : tmp = {2'h1, quant,10'h0};
 default : tmp = 16'hXXXX;
endcase
if(sig)
 alaw2linear = tmp;
 else
 alaw2linear = -(tmp);
end
endfunction
```

5.1.2 RB Selection

The PDCCH packet contains the Resource Block Map for the Downlink, this is shown in Figure 4-24. The DRBMAP (Downlink Resource Block Map) is written directly into the RB table to process the corresponding sub-frame. When the appropriate bit is set in the DRBMAP block it means the corresponding Resource Block in PDSCH packet (which is stored in PDSCH RAM content in FPGA) is to be used for this particular sub-frame and is then de-compressed and used. When the bit is not set it means there is no corresponding compressed IQ sample in the PDSCH RAM buffer to be used.

5.1.3 PDCCH Encoding

The PDCCH data buffer stores 2bit value of PDCCH data. The 2 bit value will be used to lookup a table stored in CSR, the table lookup applies the appropriate encoding and results in a 32bit value of the RE. The PDCCH data is defined in Figure 4-26.

5.1.4 PBCH Replacing

Physical Broadcast Channel replacing function will replace the appropriate resource blocks in the resource block map with the Physical Broadcast Channel that is stored in the PBCH RAM buffer, this is done at Antenna 0 & 1, symbols 3,4,5,6 at a corresponding offset. It is the expectation that Software will update the PBCH RAM buffer before the PBCH Replacing function is used. The PBCH RAM Buffer should be large enough to store the 4 symbol worth of data (3,4,5,6) which is 36*16 the format of which is detailed in Figure 4-30.

5.1.5 PSS Replacing

Similar to the PBCH Replacing the Primary and Secondary Sync Signal replacing function will replace the appropriate resource blocks with the PSS and SSS signals, this is done in Antenna 0, symbols 7,8. The PSS Replacing RAM buffers should be large enough to store 144 RE's (32b*144). The structure of which is detailed in Figure 4-31.

5.1.6 RS Replacing

The RS Replacing function will replace the appropriate resource blocks with the reference signal. It is the role of Software to ensure that the RS RAM buffer has been updated with the latest content before the RS replacing function is used. The RS replacing happens at Antenna 0 and Antenna 1, symbols 0,4,7,11 at a corresponding offset. The RAM buffer will contain the 2 bit value that will be used to do a table lookup stored in CSR in the FPGA. The result is a 32bit value to replace the original RE. The RS RAM buffer should be 64*512 (additional padding is added to simplify addressing) as shown in Figure 4-32.

5.1.7 Symbol Shift

TBD if needed.

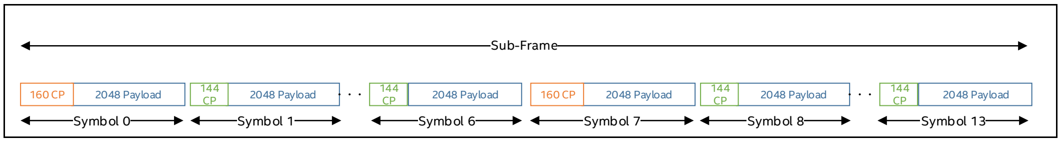
5.1.8 IFFT

When the resource block map is complete it is sent to the IFFT module for processing.

5.1.9 CP Appending

The Cyclic Prefix is added to the output of the IFFT module forming the complete LTE Symbol. Only Normal Cyclic Prefix is needed to be supported at this time. The structure of the CP is shown below. The output of this block is stored into each antenna buffer where it can be multiplexed to the CPRI interface.

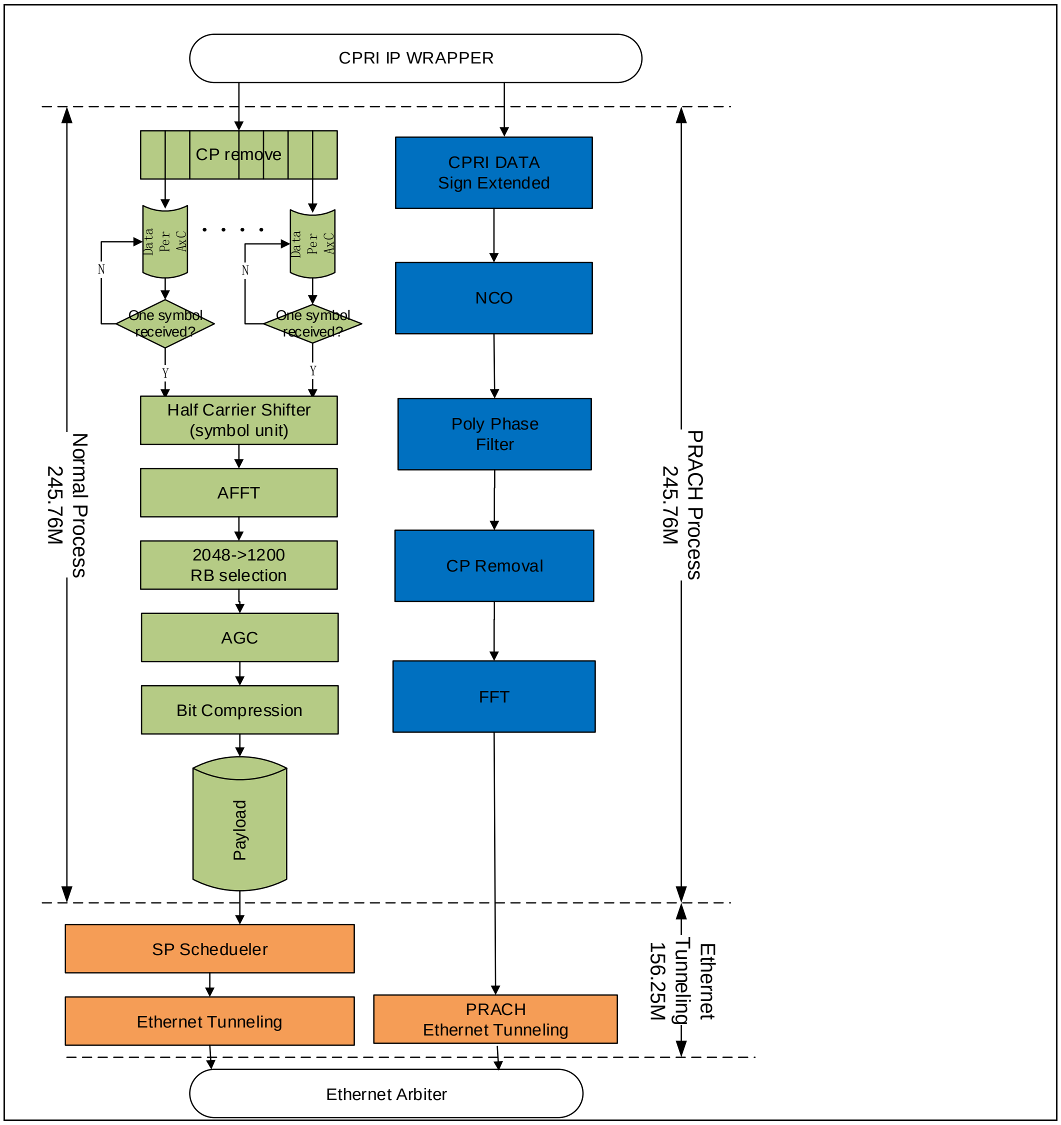
Figure 5-1. Normal Cyclic Prefix Structure



5.2 Uplink Processing

The Uplink processing in the RPE is shown below. There are two uplink paths shown, PRACH processing and PUSCH processing. In current release there is no separate PUCCH processing path, the Resource Blocks associated with PUCCH are handled as part of the PUSCH processing path and SW will process PUCCH after receiving the PUSCH packets. The PUSCH processing works on a 1ms interval and sends LTE Symbols with compressed IQ samples over Ethernet to the SW application. At the end of the Uplink processing the data is encapsulated with Ethernet header and RPE specific header and sent to the IA server.

Figure 5-2. HW Uplink Processing



5.2.1 Cyclic Prefix Removal

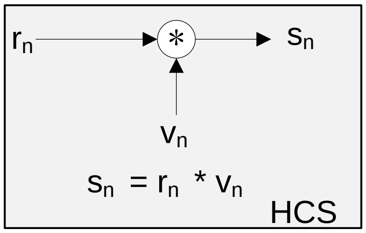
Cyclic Prefix Removal is the first stage of the PUSCH processing pipeline. Only Normal Cyclic prefix is supported in current implementation and the format is the same as Figure 5-1. The CP is removed and the rest of the samples are buffered into the Half Carrier Shift function.

5.2.2 Half Carrier Shift

The half carrier shift function works on a LTE symbol basis, hence it is required to buffer the hyper-frames received from the CPRI interface. It also is required to de-interleave the hyper-frames into separate buffers – one per antenna carrier. The main mathematics of the half carrier shift is shown as below, where rn is the payload and vn is calculated based on the below formula:

Equation 1. Half Carrier Shift

$$vn = e^{j\frac{n\pi}{N}} \quad \text{where } N=2048,\ n = 0,1\ldots2047$$



In the current implementation *vn* will be calculated in advance and stored in FPGA's ROM in the form of a table with the size of 2048x32bit = 64kbit.

Note: to keep the consistency with the process in software, the result should have some rounding operation.

Code Snippet 3. Half Carrier Shift Rounding Operation

```
result_Re =( (real * ComplexTable11[SampleIdx*2 +0]+0x4000)>>15) +
                                   (( image * ComplexTable22[SampleIdx*2 +0] + 0x4000)>>15);
result_Im = ((image * ComplexTable11[SampleIdx*2 +1] + 0x4000)>>15)+
                                     (( real * ComplexTable22[SampleIdx*2 +1] + 0x4000)>>15);
```

5.2.3 FFT

The output of the Half Carrier Shift function is used as input to the FFT, the size of which is dependent on the BW – In this case the only BW supported is 20MHz which means an FFT size of 2048 point.

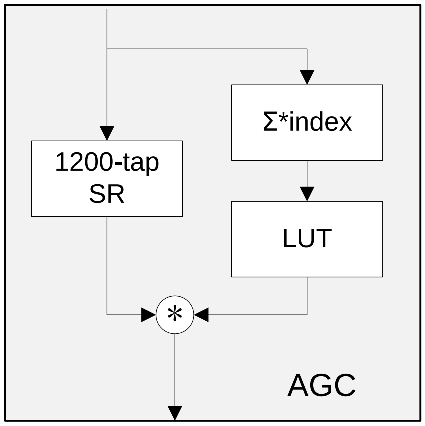
5.2.4 RB Selection

The output of the FFT is fed into the RB Selection function. The output of the FFT is 2048 samples however only the middle 1200 (1200 subcarriers) samples are useful for later PUSCH processing. The first stage of RB selection is to filter out the middle 1200 samples and discard the remaining samples. In addition to this first stage filter only some of the Resource Blocks/samples out of the 1200 samples have actually been scheduled by the upper layers. The Resource Block Map corresponding with this sub-frame would have been received by the RPE device in the previous sub-frame through the PDCCH packet. This is shown in Figure 4-25

5.2.5 AGC

The next stage in the pipeline is the AGC, as mentioned above only the useful 1200 samples will be input to this block. There're two tables in this module: one is *RB_scale* table which is indexed by *RBCnt* (The sum of all used RB in this uplink sub-frame) and the other is *AGC_scale* table which is indexed by the product of the sum of valid data and *RB_scale*. For the reason of floating point to fixed point conversion, the data in *RB_scale* table is multiply by 2^19 and the data in *AGC_scale* table is multiply by 2^8, so each product result need right shift corresponding bit after table look-up.

Figure 5-3. AGC



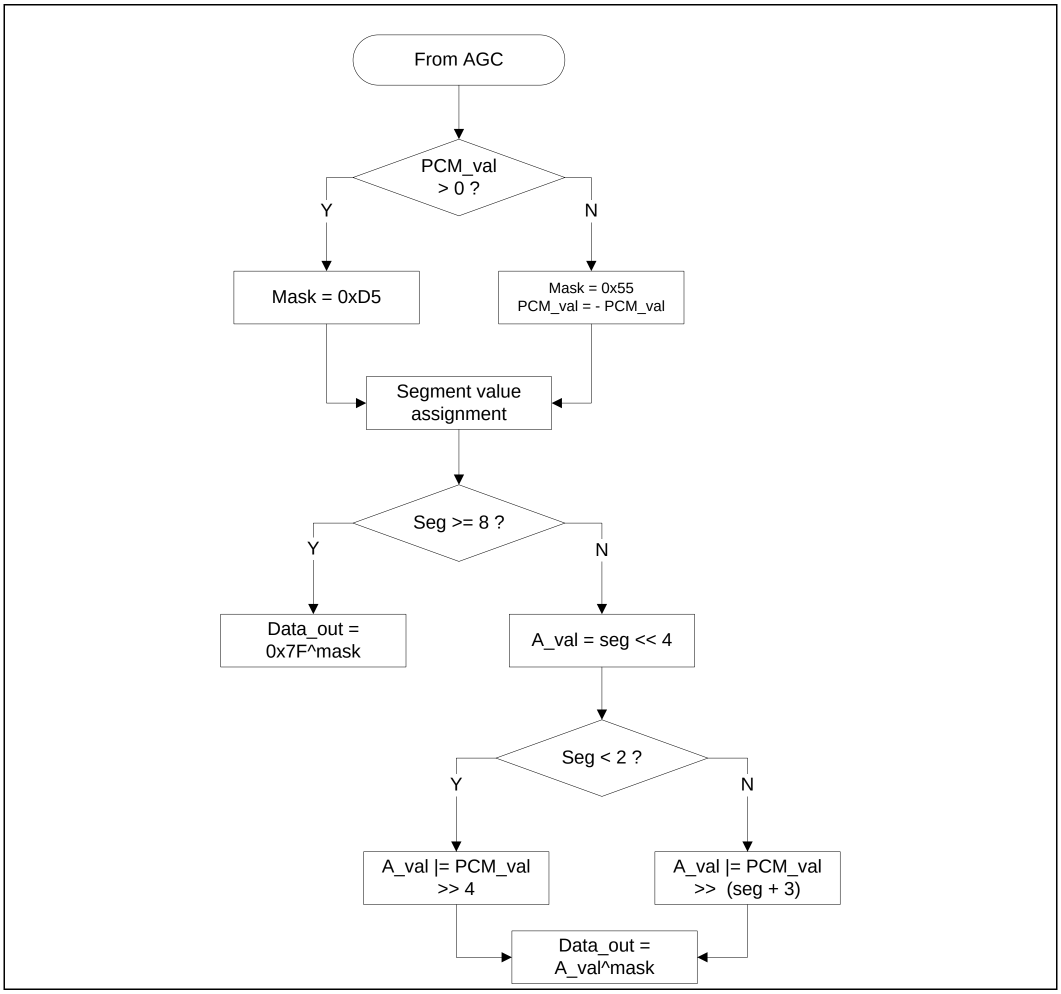
5.2.6 Compression

The compression is to convert a 16-bit linear PCM value to 8-bit A-law value, the real part and imaginary part will be compressed in parallel. The algorithm of compression is shown as below.

Code Snippet 4. Compression Algorithm

```
function [7:0] alaw;
    input signed [15:0] pcm;
    reg signed [15:0] pcm_abs;
    reg [7:0] mask;
    reg [3:0] seg;
    reg [7:0] alaw_tmp;
    begin
      if (~pcm[15]) begin
        mask = 'h00;
        pcm_abs = pcm;
end
      else begin
        mask = 'h80;
        pcm_abs = -(pcm);
      end
 if(pcm_abs[15])
  seg = 4'h8;
 else if(pcm_abs[14])
  seg = 4'h7;
 else if(pcm_abs[13])
  seg = 4'h6;
 else if(pcm_abs[12])
  seg = 4'h5;
 else if(pcm_abs[11])
  seg = 4'h4;
 else if(pcm_abs[10])
  seg = 4'h3;
 else if(pcm_abs[9])
  seg = 4'h2;
 else if(pcm_abs[8])
  seg = 4'h1;
 else
  seg = 4'h0;
      if (seg >= 'h8) alaw = mask ^'h7F;
      else begin
        alaw_tmp = seg << 'h4;
if (seg < 'h2) alaw = mask ^ (((pcm_abs >> 4) & 'hf) | alaw_tmp);
        else alaw = mask ^ (((pcm_abs >>(seg+'h3))&'hf) | alaw_tmp);
      end
    end
endfunction // alaw
```

Figure 5-4. Compression Block Diagram



5.3 PRACH Process

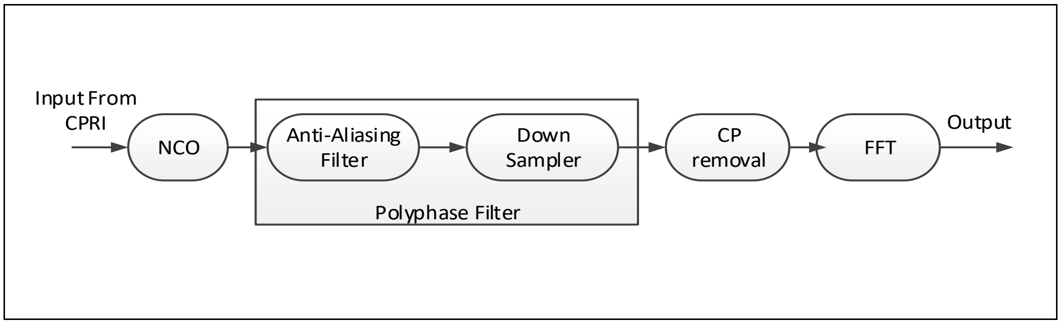
The PRACH block implements extracting PRACH from the received IQ Samples and does not include detection of appropriate preambles for PRACH. PRACH will be extracted in the time domain using the following hierarchy of blocks

a) NCO: frequency down conversion to shift the band of PRACH to lowest frequency band and half sub carrier frequency shift

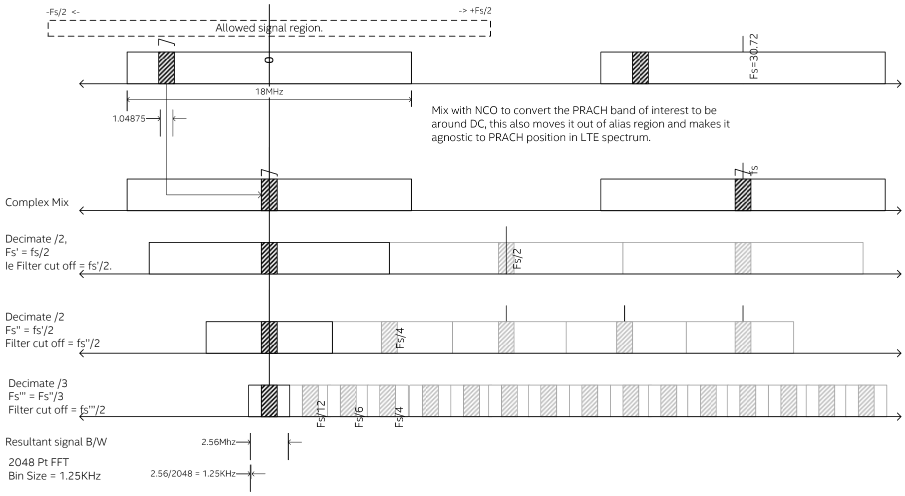
b) Poly Phase filter including down sampling to get to desired FFT point size

c) FFT on the output of the down sampler.

Table 5.1



Spectral Overview with 20MHz bandwidth



Mix with NCO to convert the PRACH band of interest to be around DC, this also moves it out of alias region and makes it agnostic to PRACH position in LTE spectrum.

I suspect the FFT shift function, which is in effect a conjugation, is used to remove the spectral inversion resulting from odd decimation.How? But we are dealing with a complex signal.?

1.1.1.1.1 PRACH Format Selection Criteria

Table 5.2 describes the criteria for implementing PRACH functionality on sub frames. This definition in line with 3GPP specification 36.211 section “5.7 Physical Random Access Channel” Table 5.7.1-2.

Table 5.2 PRACH packets within sub-frames

```
function config_sf_mask_s map_config_index( int index );
    // Define which sub-frames are to be sent to the model
    // This is derived from Table 5.7.1-2 ETSI TS 136 211 V10.0.0 (2011-01)
```

```
      // For values > 15, no sub-frames are processed.
      config_sf_mask_s return_mask;
      bit[10:0] sub_frame_mask;
      case( index )
        00: sub_frame_mask = {`PRACH_MON_EVEN_ONLY,  10'b00_00_00_00_10 };
        01: sub_frame_mask = {`PRACH_MON_EVEN_ONLY,  10'b00_00_01_00_00 };
        02: sub_frame_mask = {`PRACH_MON_EVEN_ONLY,  10'b00_10_00_00_00 };
        03: sub_frame_mask = {`PRACH_MON_ANY,        10'b00_00_00_00_10 };
        04: sub_frame_mask = {`PRACH_MON_ANY,        10'b00_00_01_00_00 };
        05: sub_frame_mask = {`PRACH_MON_ANY,        10'b00_10_00_00_00 };
        06: sub_frame_mask = {`PRACH_MON_ANY,        10'b00_01_00_00_10 };
        07: sub_frame_mask = {`PRACH_MON_ANY,        10'b00_10_00_01_00 };
        08: sub_frame_mask = {`PRACH_MON_ANY,        10'b01_00_00_10_00 };
        09: sub_frame_mask = {`PRACH_MON_ANY,        10'b00_10_01_00_10 };
        10: sub_frame_mask = {`PRACH_MON_ANY,        10'b01_00_10_01_00 };
        11: sub_frame_mask = {`PRACH_MON_ANY,        10'b10_01_00_10_00 };
        12: sub_frame_mask = {`PRACH_MON_ANY,        10'b01_01_01_01_01 };
        13: sub_frame_mask = {`PRACH_MON_ANY,        10'b10_10_10_10_10 };
        14: sub_frame_mask = {`PRACH_MON_ANY,        10'b11_11_11_11_11 };
        15: sub_frame_mask = {`PRACH_MON_EVEN_ONLY,  10'b10_00_00_00_00 };
        default: sub_frame_mask = {`PRACH_MON_ANY,   10'b00_00_00_00_00 };
      endcase
      return_mask.all_sframes = sub_frame_mask[10];
      return_mask.sframe_mask = sub_frame_mask[9:0];
      return( return_mask );
    endfunction: map_config_index
```

Table 5.3 PRACH NCO calculations

The register setting of PRACH_NCO_PHI_INC is a function of the where the UE puts the PRACH sequence in the spectrum. This “PRACH frequency offset” is specified by System Information Block 2 (SIB-2) by the eNB to the UE, as the first resource block used by the PRACH (nRB).

The location of the PRACH is defined in section 5.7.3 "Baseband signal generation" of 3GPP specification 36.211. This defines the encoding of the PRACH where here the decode is defined:

So it can be said that:

$$NCO_{phi\_inc} = f(nRB)$$

This function is defined below:

$$nRB = n_{PRB}^{RA} = first\ resource\ block\ (RB)\ by\ PRACH$$

$$0 \leq nRB \leq 94$$

The NCO frequency (f_{NCO}) required to mix down the PRACH to be DC centred is:

$$f_{NCO} = \left[\varphi + K\left(k_0 + \frac{1}{2}\right) + floor\left(\frac{N_{zc}}{2}\right)\right] \times \Delta f_{RA}$$

Where

$$N_{zc} = length\ of\ PRACH\ sequence = number\ of\ PRACH\ bins\ used = 839$$

$$K = \frac{\Delta f}{\Delta f_{RA}} = \frac{OFDM\ freq\ bin\ width}{PRACH\ freq\ bin\ width} = \frac{15kHz}{1.25kHz} = 12$$

$$k_0 = First\ OFDM\ subcarrier\ used\ by\ PRACH$$

$$-600\ \leq k_0 \leq 600$$

$$k_0 = n_{PRB}^{RA} N_{sc}^{RB} - N_{RB}^{UL} N_{SC}^{RB}/2$$

$$N_{SC}^{RB} = number\ of\ OFDM\ subcarriers\ in\ a\ RB = 12$$

$$N_{RB}^{UL} = number\ of\ RB\ in\ a\ symbol = 100$$

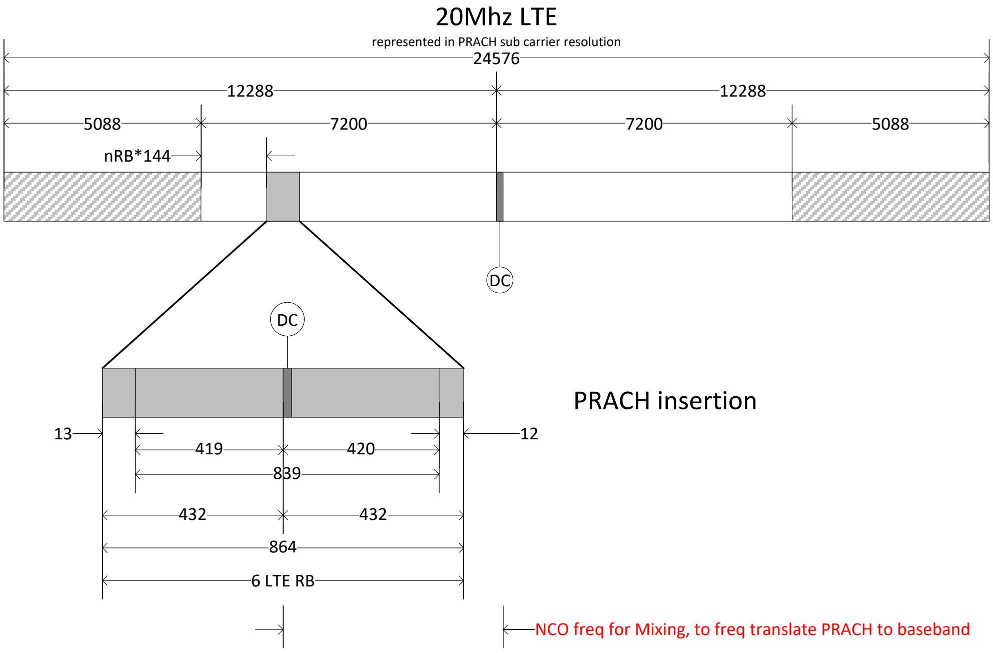
$$\varphi = 7$$

Note: The $floor\left(\frac{N_{zc}}{2}\right)$ term does not appear in 3GPP specification 36.211 because it assumes that the start of the PRACH sequence is at DC, whereas here the middle of the PRACH sequence is required to at DC.

Note: The φ value together with the K(1/2) gives an offset of 13 PRACH subcarriers. This accounts for the standards 13 offset at the front of the 839 PRACH sequence in the 864 subcarriers allocated for PRACH in six resource blocks.

The PRACH_NCO_PHI_INC register setting (NCO_{phi_inc}) is:

$$NCO_{phi\_inc} = f_{NCO} \times \frac{2^{32}}{f_{sample}}$$


20Mhz LTE
represented in PRACH sub carrier resolution
24576
12288
12288
5088
7200
7200
5088
nRB*144
DC
DC
PRACH insertion
13
12
419
420
839
432
432
864
6 LTE RB
NCO freq for Mixing, to freq translate PRACH to baseband

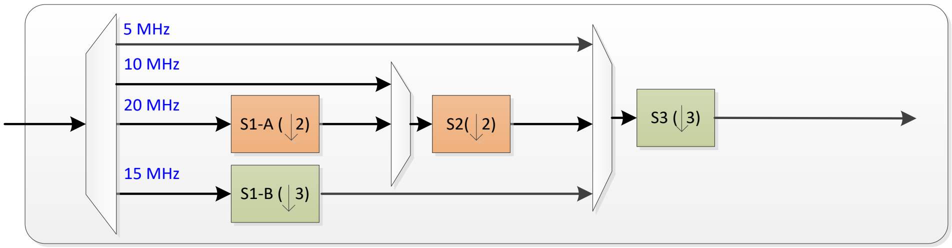
5.3.1 NCO and mixer

The received signal is multiplied with complex sine and cosine and used for down conversion of the signal to base band. The NCO output has 18 bit precision. The mixer has starting offset based on the preamble format which indicates start of multiplication of received sampled with NCO. The NCO frequency is configurable.

5.3.2 Poly Phase Filter

After the data is passed through an anti-aliasing filter it is passed through a down-sampling block. The down sampling is done using 4 filters spread out over 3 stages in order to decimate all the PRACH channels into 2K samples for Formats 0-3 in all supported bandwidths. Format 4 PRACH channels are decimated into 1K points except for 10MHz and 20MHz where they can be decimated to 2K points:

Figure 5: Poly Phase Filter



The ↓3 decimation filter has 51 taps and the ↓2 filter has 25 taps. Below are the resulting PRACH FFT sizes for the supported frequency bandwidth sampling.

Table 5.4 FFT Size for Preamble Format

Preamble Format	Bandwidth			
	5 MHz	10 MHz	15 MHz	20 MHz
0 & 1	2048	2048	2048	**2048**
2 & 3	2 x 2048	2 x 2048	2 x 2048	2 x 2048
4	1024	2048 or 1024	1024	1024

Note: This release only supports Format 0 with 20MHz bandwidth.

1.1.1.1.1.1 Filter Coefficients

The fixed filter coefficients are listed in **Error! Reference source not found.**:

Table 5.5 Fixed PRACH filter coefficients

```
#define  ZERO_FILTER_COEFF                 0
#define  FIRST_FILTER_COEFF                13 /*25*/
#define  SECOND_FILTER_COEFF               13 /*25*/
#define  THIRD_FILTER_COEFF                26 /*51*/
```

```
static const Cpa16U firstFilterIndex[FIRST_FILTER_COEFF*2] =
{
    0,65300,0,446,0,64674,0,1592,0,62375,
    0,10322,16384,10322,0,62375,0,1592,0,64674,
    0,446,0,65300,0
};

static const Cpa16U secondFilterIndex[SECOND_FILTER_COEFF*2] =
{
    0,65300,0,446,0,64674,0,1592,0,62375,
    0,10322,16384,10322,0,62375,0,1592,0,64674,
    0,446,0,65300,0
};

static const Cpa16U thirdFilterIndex[THIRD_FILTER_COEFF*2] =
{
    855,382,65219,65363,65210,314,320,297,65177,65063,
    65252,457,657,279,64904,64625,65255,948,1338,283,
    63894,63195,65251,4374,9148,11208,9148,4374,65251,63195,
    63894,283,1338,948,65255,64625,64904,279,657,457,
    65252,65063,65177,297,320,314,65210,65363,65219,382,
    855
};
```

5.3.3 Cyclic Prefix Removal

In LTE, the received OFDM symbol consists of cyclic prefix added during transmission for eliminating ISI (Inter Symbol Interference). ISI is a particular issue with PRACH because UEs are using the PRACH mechanism before any timing advance is applied by the UE to allow for its distance from the BBU. This block removes the cyclic prefix.

Note that the PRACH Cyclic Prefix removal is completely independent of PUSCH CP removal.

1.1.1.1.2 FFT

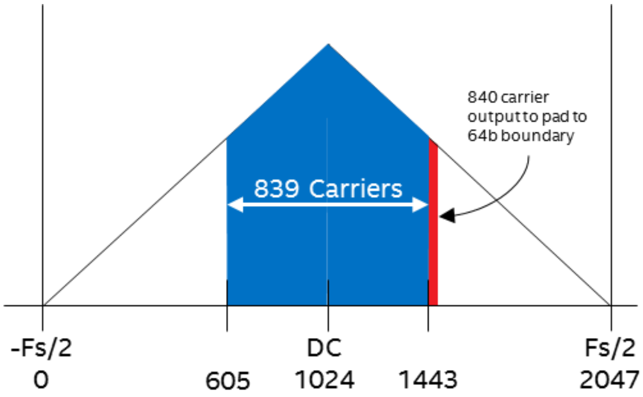
The FFT is 2048 point only; as required by 20MHz Format 0.

The FFT uses the Altera Mega Core IP. The FFT is set to use dynamic scaling. The dynamic scaling determined by the block should be passed forward in the PRACH packet.

The samples will be output in natural frequency bin order.

At 20MHz bandwidth and after the down sampling each sub carrier has a width of 1.25k. Only the center 839 sub-carriers are of the 2048 output by the FFT are used in the PRACH process. To allow alignment to a 64b boundary 840 carriers are output from the blocks as shown in the figure below.

Table 5.6 Spectrum used by PRACH



6 Additional Interfaces

6.1 MXG

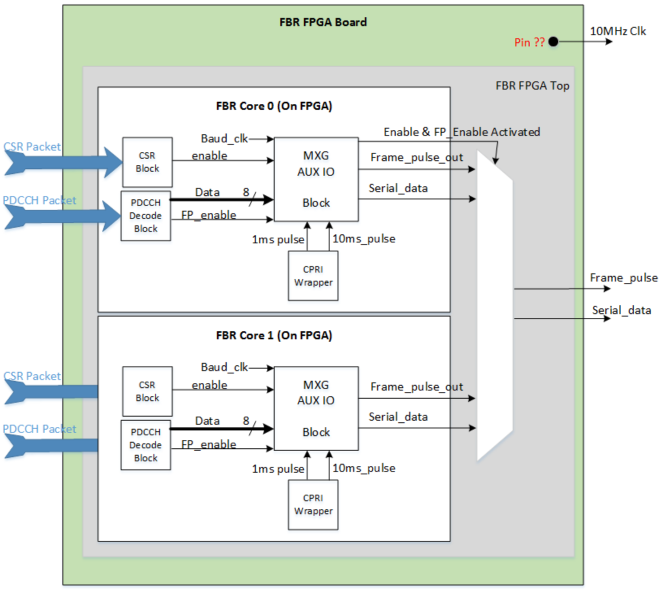
6.1.1 Functional Description

The MXG feature is specifically implemented to communicate information from within the FerryBridge design to the outside world via auxiliary data pins. The functional description and features of the MXG equipment can be viewed in the links provided in **Error! Reference source not found.**.

In terms of the FBR functional implementation of the feature, the following diagram highlights the overall design feature and 3 serial pins used to connect FBR to the MXG equipment, namely:

- 10MHz clock
- Frame_pulse
- Serial_data

Figure 6: MXG feature implementation inside Ferry Bridge



The following set of features are implemented inside the FBR design:

- Dedicated Mode of operation
  - A serial data transmission format, working at a baud rate of 460800
  - Polarity will be positive only
  - 8 bit transmission character defined as follows:
    - Bits (b7 b6 b5 b4 b3 b2 b1 b0, b0),
    - Bit b0 is least-significant bit

        - Bit b7 is most significant bit
        - The signaling transmission consists of a start bit ‘0’, followed by the 8 data bits (transmitted LSB first) followed by a stop bit ‘1’

- 10MHz Clock
    - Provide a 10MHz Clock Signal from the FBR board to external world

- Frame Pulse
    - Provide a 10ms Frame Pulse signal from the FBR board to the external world

- FBR Register to enable the Feature
    - MXG_AUX_IO_ENABLE

- PDCCH packet to be used by Software to hold the contents of the 8 bit character that is to be sent serially & 1 bit FP_ENABLE that is used to trigger the FP_PULSE bit
    - SW will update the 8 bit data register every 1ms using PDCCH packet only
    - The format of the data inside the PDCCH packet will be:
        - Bits (b7 b6 b5 b4 b3 b2 b1 b0, b0),
        - Bit b0 is least-significant bit
        - Bit b7 is most significant bit
    - SW will only ever write to the FP_ENABLE bit once (see assumptions on this later in this document)

- The FBR design has two cores. One MXG feature is available per core. To save pin resources only one MXG data can be accessed at time either the one within core0 or within core1. In a case were both MXG features would be enabled at the same time the MXG in core0 will take precedence over the MXG in core1. So to enable the MXG in core1 be sure to disable first the MXG function in core0.

6.1.2 FBR Design Implementation of the AUX IO stream

How the Feature is intended to work in conjunction with Software the the MXG equipment itself…

Step 1. Software will write to the CSR Register MXG_AUX_IO_ENABLE in order to switch on the function, determine the baud rate and the endianness of the data transmission. Please refer to section **Error! Reference source not found.** for more information on this register.

Step 2. Software will then write the AUX PDCCH packet AUX[7:0] - the data stream can be any desired value as determined by SW. The FP bit in the PDCCH packet should be set to 1’b1. Note that all future PDCCH packets will have the FP bit set to 1’b0, such that the FP_PULSE will not be written out continuously on a 10ms cycle.

Step 3. With the feature enabled and the FP_PULSE requested to be sent, the FBR design will output the FP_PULSE once it recognizes its next internal 10ms pulse. It will then attempt to output the serial data stream as soon as the first positive edge of the baud_clk is found. Note that the alignment of the FP_PULSE and the first START_BIT of the serial data stream will not be in exact alignment with the positive edge of FP_PULSE, it is expected that the FP_PULSE will occur and then soon after that point in time the serial data will begin transmission.

Step 4. For each and every internal 1ms pulse, the FBR design will output the contents of its internal AUX_SERIAL_DATA[7:0] register. Note that this register will be updated every time a new PDCCH packet (AUX[7:0]) arrives into the FBR design.

Step 5. The transmission of the serial data stream will continue to be output as long as the CSR register enable bit is set (MXG_AUX_IO_ENABLE)

6.1.3 10 MHz Clock

The external LMK component to FBR (located on FBR Board itself) can generate the 10MHz output on its CLK1 output. This is input to pins F9/G9 on the FPGA Board (Not to SMA connector). This 10MHz will be Frequency locked to the CPRI reference clock and output to the MXG equipment.

6.1.4 Frame Pulse

In terms of the 10ms Frame Pulse being transmitted out of the FBR design, it is s a requirement that software will only ever assert this pin once via the PDCCH packet communication as this is used to kick start the MXG alignment.

6.1.5 Alignment of the Serial Data Transmission with the Frame Pulse

In terms of aligning the serial data transmission with the Frame Pulse, we transmit the 'Start bit' on the 1st positive edge of the clock driving the data out (@ baud rate 460800) as soon as the Frame Pulse signal is edge detected.

6.1.6 New Pin Assignments on FerryBridge Board

Connector	Pin	Signal
J3	Pin 2	Serial data IO stream
J3	Pin 3	Frame Pulse stream
J3	Pin 6	10MHz clock