KeyStone Architecture
Bit Rate Coprocessor (BCP)

User's Guide

Literature Number: SPRUGZ1A
August 2011–Revised May 2015

Contents

List of Figures

List of Tables

Preface

About This Manual

Wireless base stations currently use cost-efficient DSPs for their implementations. The system cost of the base station is a function of the number of channels that the DSP can support. As the wireless systems are converted from second generation to third and fourth generation systems, the system complexity has increased by an order of magnitude. These wireless systems require an increasing amount of baseband bit processing. A very cost effective and synergistic solution has been designed using a DSP and Bit Rate Coprocessor (BCP). This manual describes the architecture and use of the BCP peripheral.

This document uses the following conventions:

- Commands and keywords are in **boldface** font.
- Arguments for which you supply values are in *italic* font.
- Terminal sessions and information the system displays are in screen font.
- Information you must enter is in **boldface screen font**.
- Elements in square brackets ([ ]) are optional.

Notation Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labelled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.
- The term “word” describes a 32-bit value.

Notes use the following conventions:

NOTE: Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

CAUTION

Indicates the possibility of service interruption if precautions are not taken.

WARNING

Indicates the possibility of damage to equipment if precautions are not taken.

Related Documentation from Texas Instruments

C66x CorePac User Guide	SPRUGW0
C66x CPU and Instruction Set Reference Guide	SPRUGH7

Introduction

NOTE: The information in this document should be used in conjunction with information in the device-specific Keystone Architecture data manual that applies to the part number of your device.

Topic **Page**

1.1 Purpose of the Peripheral

The Bit Rate Coprocessor (BCP) is a programmable peripheral for baseband bit processing. Integrated into the Texas Instruments DSP, it supports FDD LTE, TDD LTE, WCDMA, TD-SCDMA, HSPA, HSPA+, WiMAX 802.16-2009 (802.16e), and monitoring/planning for LTE-A. It accelerates physical layer functions that generally have little or no customer intellectual property and either consume many cycles when implemented on a processor core or enable a smoother data flow or use-model. The BCP accelerates the Layer 1 downlink processing from transport blocks down to OFDM symbols (LTE/WiMAX) or physical channel data (WCDMA/TD-SCDMA, HSPA/+). It accelerates Layer 1 uplink processing from OFDM symbols to transport blocks for LTE/WiMAX (with the help of Turbo and convolutional decoder accelerators) or from physical channel data to the output of rate dematching or LLR combining for WCDMA/TD-SCDMA, HSPA/+.

1.2 Features

Primary functionalities of the BCP peripheral include the following:

- CRC
- Turbo / convolutional encoding
- Rate Matching (hard and soft) / rate de-matching
- LLR combining
- Modulation (hard and soft)
- Interleaving / de-interleaving
- Scrambling / de-scrambling
- Correlation (final de-spreading for WCDMA RX and PUCCH correlation)
- Soft slicing (soft demodulation)
- 128-bit Navigator interface
- Two 128-bit direct I/O interfaces
- Runs in parallel with DSP
- Internal debug logging

1.3 Abbreviations

Table 1-1. Abbreviations

Term	Definition
3GPP	3rd Generation Partnership Project
BCP	Bit Rate Coprocessor
CDMAHP	Deprecated name for Packet DMA (PKTDMA)
COR	Correlation submodule
CPPI	Deprecated name for Navigator
CRC	Cyclic redundancy check submodule
DIO	Direct I/O submodule
DNT	De-interleaving (for WCDMA) submodule
ENC	Turbo and convolutional encoding submodule
INT	Interleaving (for WCDMA) submodule
LLR	Log Likelihood Ratio
LTE	Long Term Evolution
MOD	Modulation submodule
MMR	memory mapped register
PKTDMA	Packet DMA (previously named CDMAHP)
RD	Rate De-matching submodule
RM	Rate matching submodule
SSL	Soft slicing submodule
TAC	Transmit Accelerator
TCP3D	Turbo decoder Coprocessor 3
TM	Traffic manager submodule
WiMAX	Worldwide Interoperability for Microwave Access

Architecture

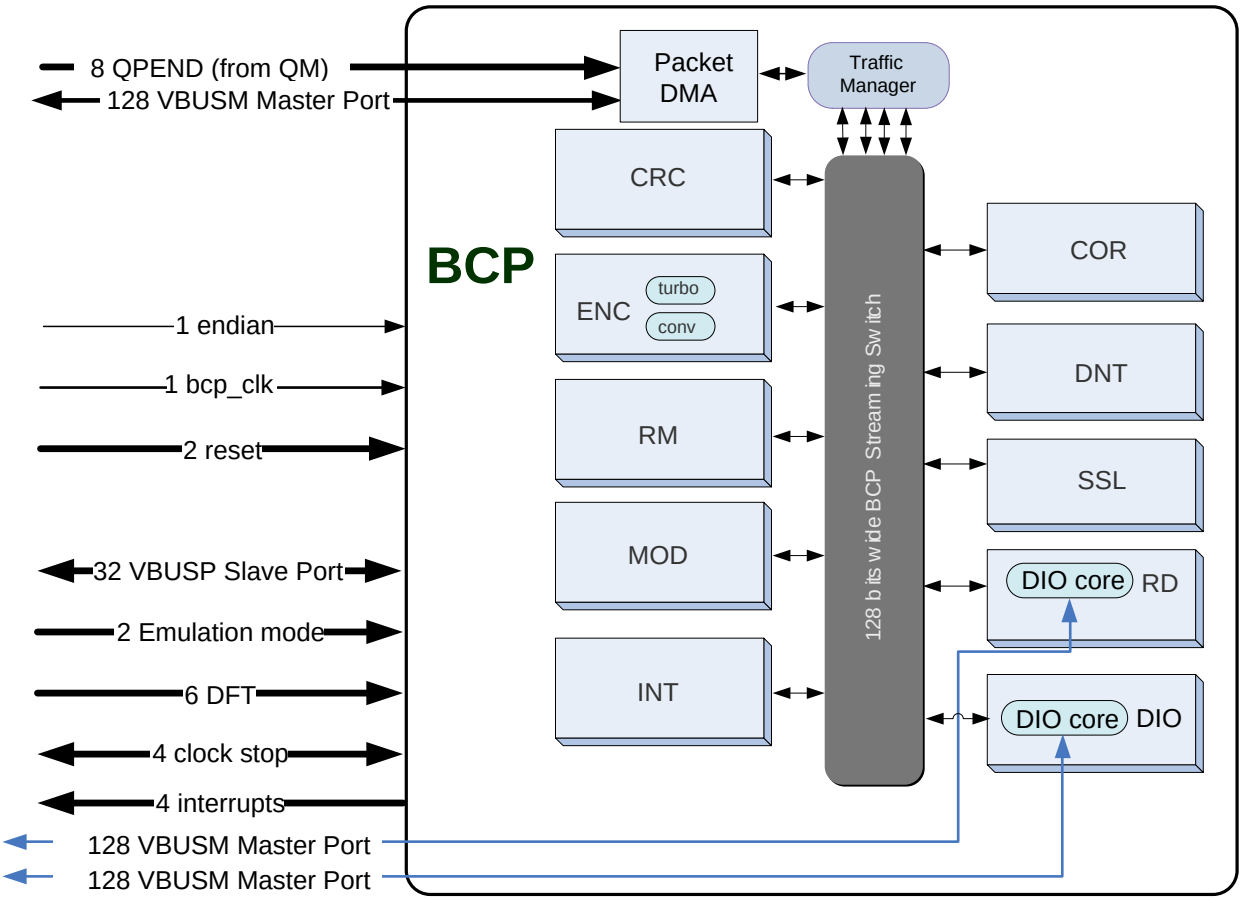
This chapter describes the details of the architecture of the peripheral and how it is structured.

Topic **Page**

2.1 Block Diagram

The block diagram of the BCP is shown in Figure 2-1.

Figure 2-1. BCP Block Diagram



Each sub-module is named based on its functionality as follows:

- CDMAHP – Navigator interface
- COR – Correlation
- CRC – Cyclic redundancy check
- DIO – Direct I/O
- DNT – De-interleaving (for WCDMA)
- ENC – Turbo and convolutional encoding
- INT – Interleaving (for WCDMA)
- MOD – Modulation
- RD – Rate De-matching
- RM – Rate matching
- SSL – Soft slicing
- TM – Traffic manager

The interrupt lines allow errors or other debug events to trigger interrupts for any combination of the DSP cores. There are three VBUSM master ports: one for the Navigator interface and two for the direct I/O interfaces. The configuration bus is used to read and write registers inside the BCP and to configure the Navigator interface. The queue pend signals are used to activate the processing via the Navigator interface. The BCP clock is designed to be a CPU/3 clock – nominally at 400MHz when the DSP clock is 1.2 GHz. The DFT (design for test) inputs are for internal (TI) use. The clock stop set of signals tells the BCP that the clock will be stopped (for power savings) after the BCP acknowledges the request. Emulation mode is used when debugging the device from an emulator; it allows the BCP to stop when a breakpoint is hit. The BCP supports big and little endian modes.

2.2 BCP Streaming Switch

Data is transferred from one submodule to another inside the BCP via the BCP Streaming Switch (BSS). It is a non-blocking switch fabric that directs data from one processing submodule to the next. When a process is finished on one submodule, the data is forwarded to the next submodule for processing. The path through the BCP is programmed by the DSP for each data packet processed via header information at the beginning of that packet. A packet can follow any path within the bounds of the connectivity matrix shown below. Masters are listed in the first column and slaves are listed across the first row. Wherever a master has a direct path to a slave there is a table entry of “1.” In general, there are two logical paths of processing in the BCP, one mainly for uplink and the other mainly for downlink and uplink interference cancellation data reconstruction.

The shaded part of Table 2-1 represents the uplink path and the non-shaded is the downlink.

Table 2-1. BCP Streaming Switch Connections

	TM	DIO	CRC	ENC	RM	MOD	INT	DNT	RD	SSL	COR
TM	1[(1)]	1	1	1	1	1	1	1	1	1	1
DIO	1		1	1	1	1	1	1	1	1	1
CRC	1	1		1			1				
ENC	1	1			1		1				
RM	1	1				1	1				
MOD	1	1									
INT	1	1			1						
DNT	1	1							1		
RD	1	1						1			
SSL	1	1						1	1		
COR	1	1								1	

(1) For loopback

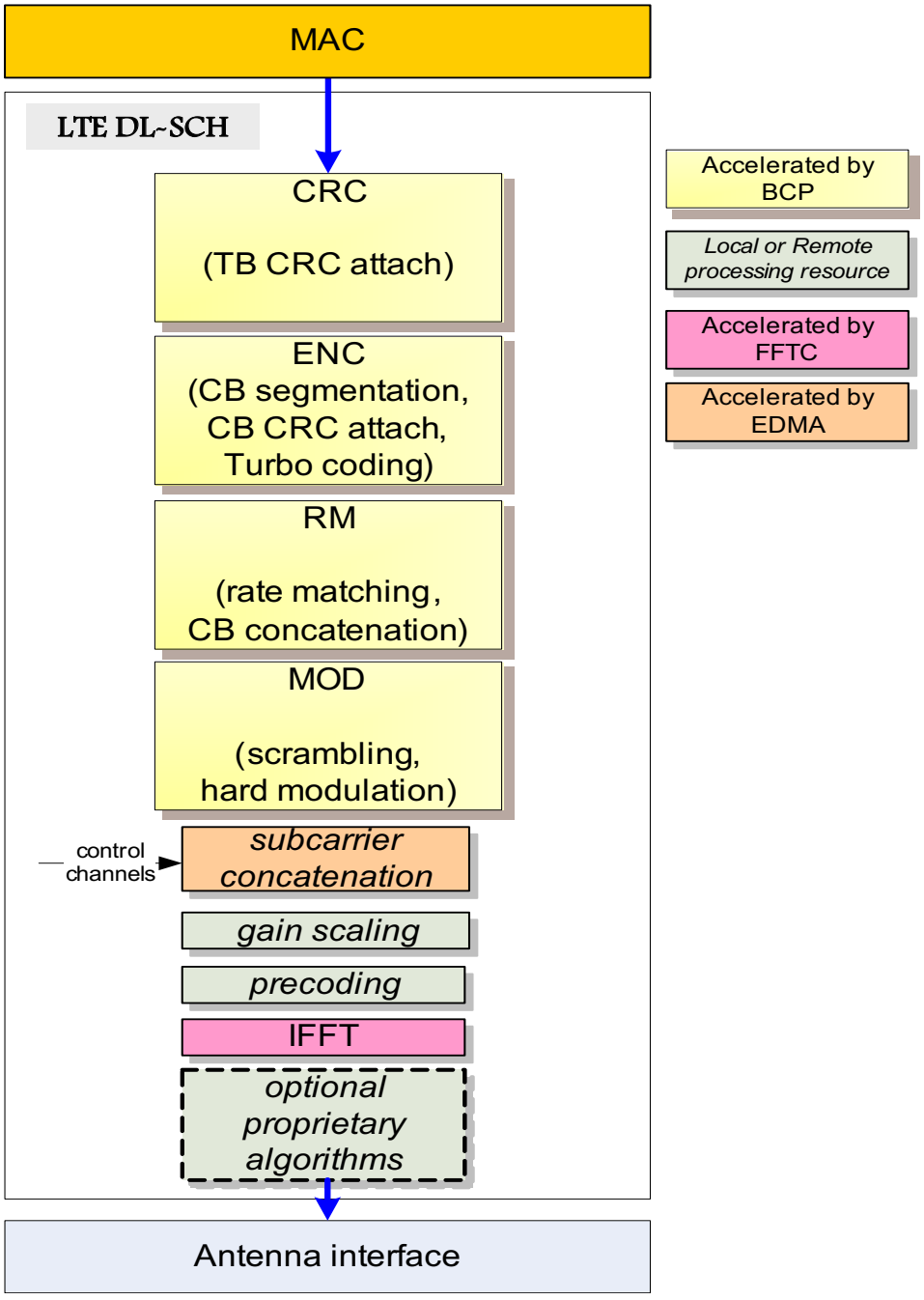
All submodules can be used stand-alone or can effectively be bypassed because all submodules have an input and output path to the traffic manager (TM).

2.3 Use-Case Examples

Example flow diagrams for the main BCP use-cases are given in the following sections. The partitioning of functionality among the submodules is also shown.

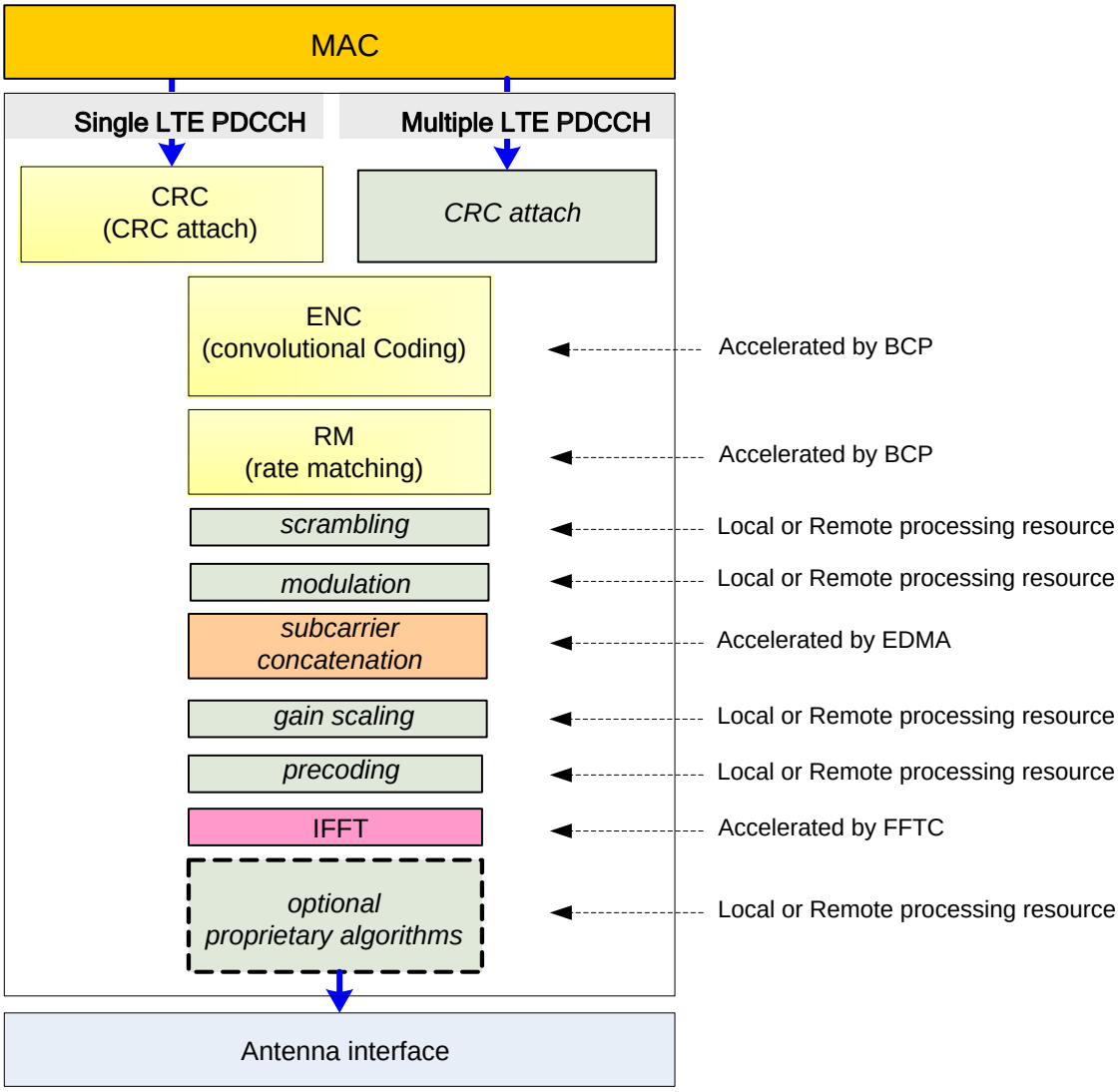
2.3.1 LTE DL-SCH

Figure 2-2. Example LTE DL-SCH Flow Diagram



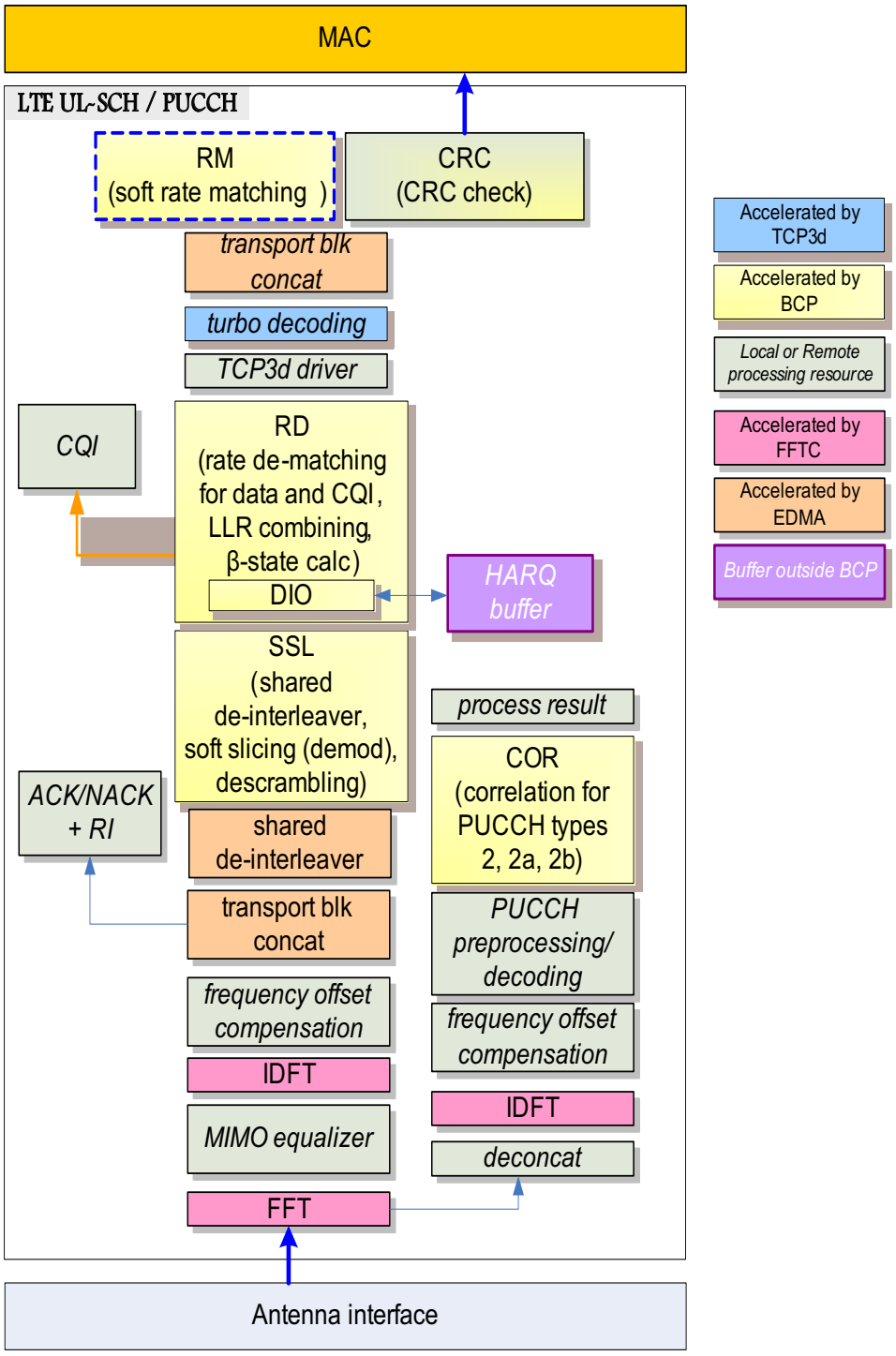
2.3.2 LTE PDCCH

Figure 2-3. Example LTE PDCCH Flow Diagram



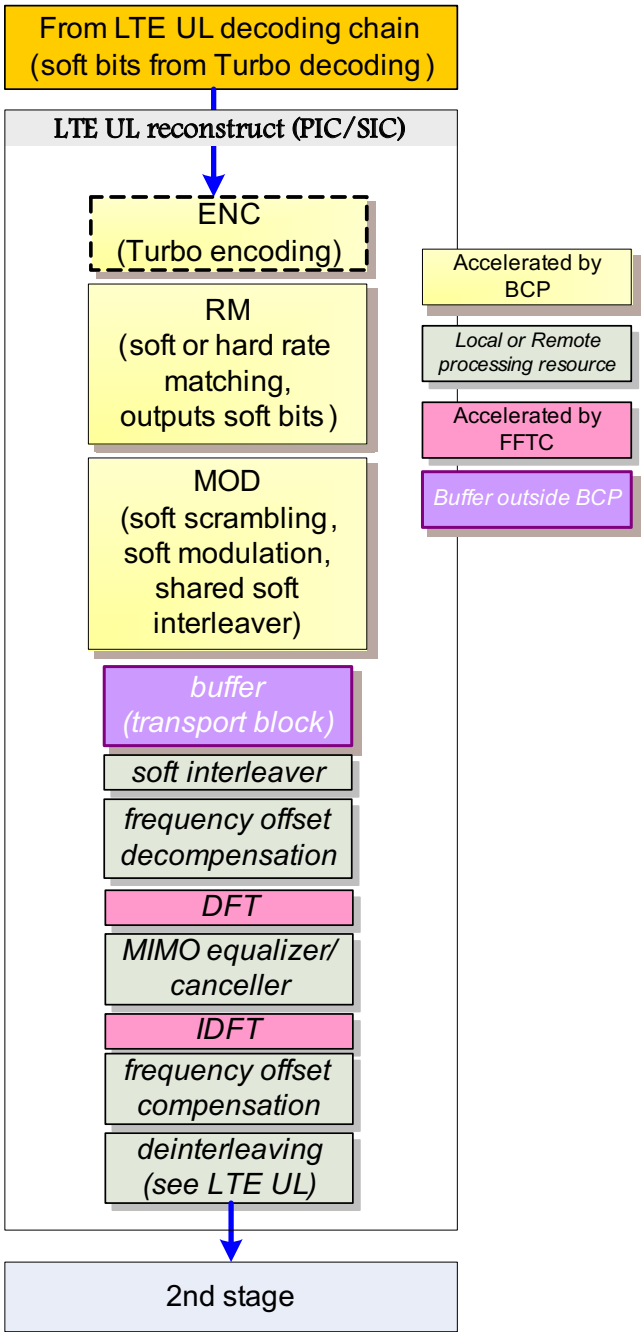
2.3.3 LTE UL-SCH / PUCCH

Figure 2-4. Example LTE UL-SCH / PUCCH Flow Diagram



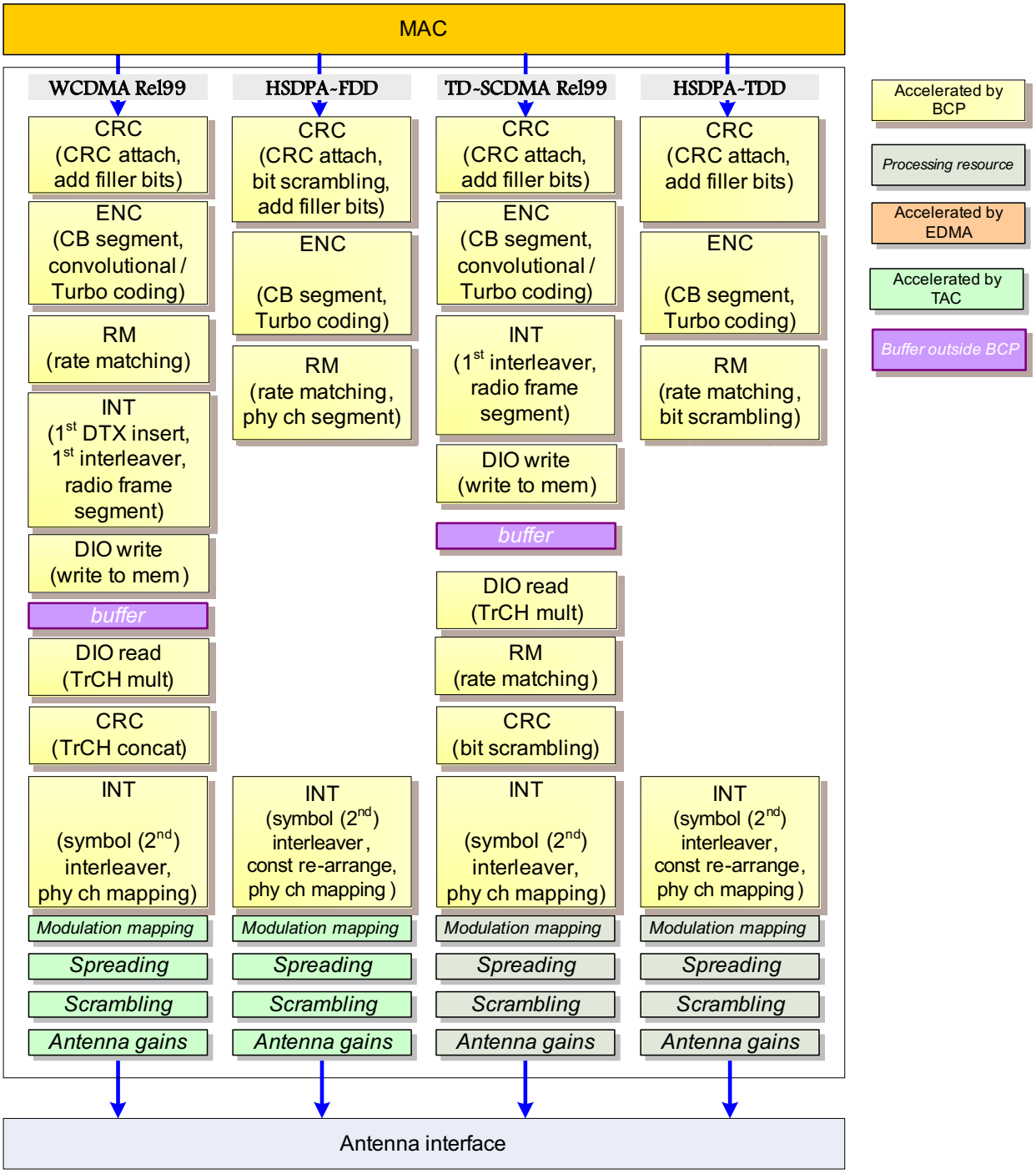
2.3.4 LTE PIC/SIC

Figure 2-5. Example LTE PIC / SIC Flow Diagram



2.3.5 WCDMA / TD-SCDMA Downlink

Figure 2-6. Example WCDMA / TD-SCDMA Downlink Flow Diagram

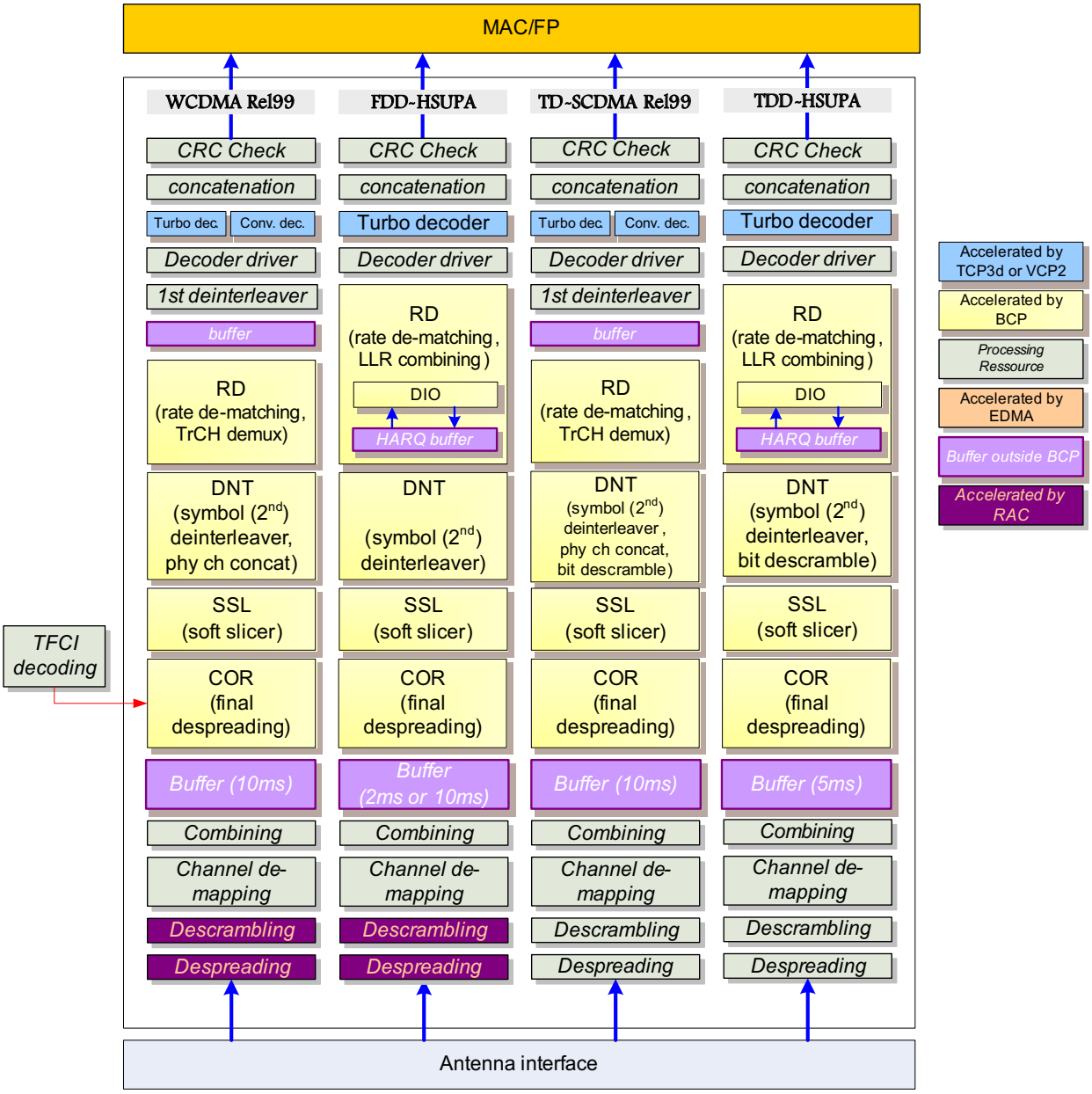


Note that the Rel-99 downlink flow diagram shows an external buffer that breaks the BCP processing between the upper transport channel processing and the lower CCTrCH processing. In this scenario, each transport channel would be submitted via a separate job (packet) for the upper transport channel processing stage. The lower CCTrCH processing would then be accomplished by sending one job per radio frame for that CCTrCH. This division of processing requires that the upper transport channel processing be complete and written to memory before the associated CCTrCH processing starts by reading the input data back into the BCP. If there are enough users active and the transport channel jobs and CCTrCH jobs are submitted in the same order, the inherent latency of the tasks will cause the requirement to be met automatically. Otherwise, the “flush” packet indicator can be used to force the BCP to wait to read the CCTrCH processing input data until the transport channel processing has completed and written its output data to memory. (Normally, a dummy “drop” packet would be required immediately after the “flush” packet to guarantee that the Packet DMA in the BCP did not read the second stage processing data too early. However, since the DIO submodule is used instead of the Packet DMA to write the output data from the 1st stage and read the input data for the second stage, the “flush” packet by itself is sufficient.)

The Rel-99 uplink processing only requires CCTrCH processing jobs since the BCP does not accelerate the transport channel processing (first de-interleaver).

2.3.6 WCDMA / TD-SCDMA Uplink

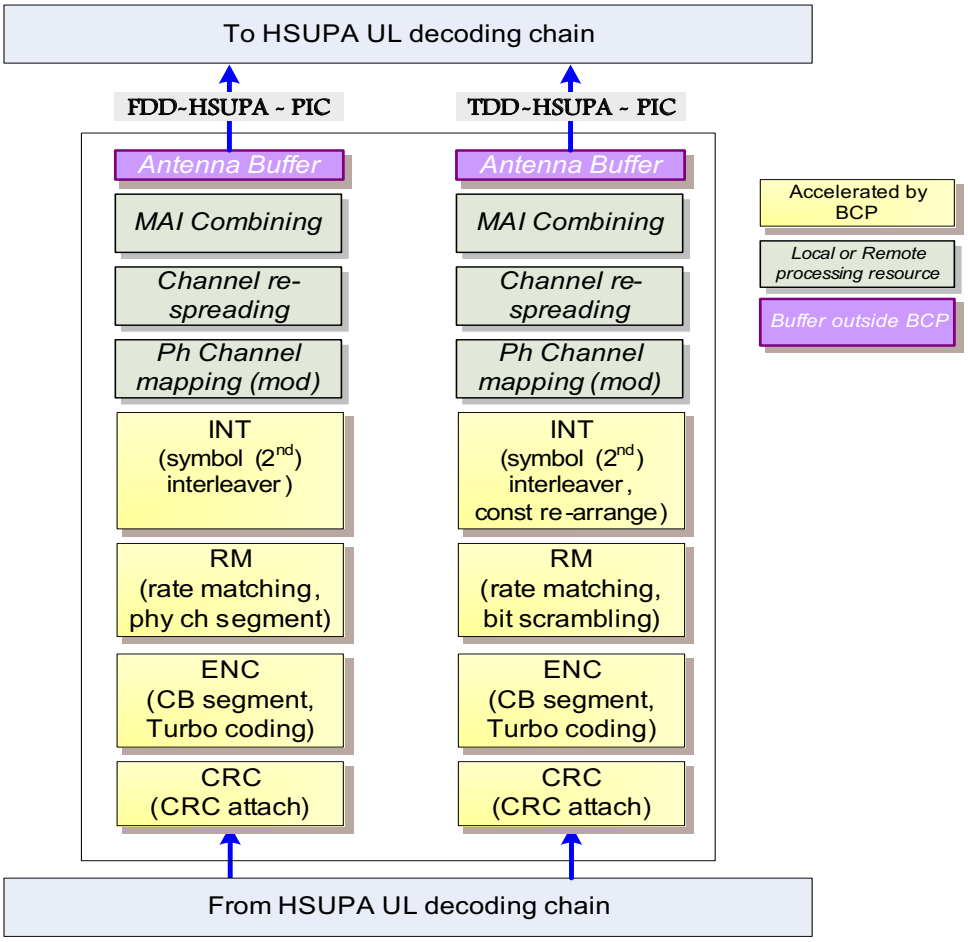
Figure 2-7. WCDMA / TD-SCDMA Uplink Flow Diagram



Note that the SSL submodule does not support 8-PSK modulated data for TD-SCDMA Rel-99 uplink. The 8-PSK modulated data should be demodulated by software before sending it to the BCP DNT and RD submodules.

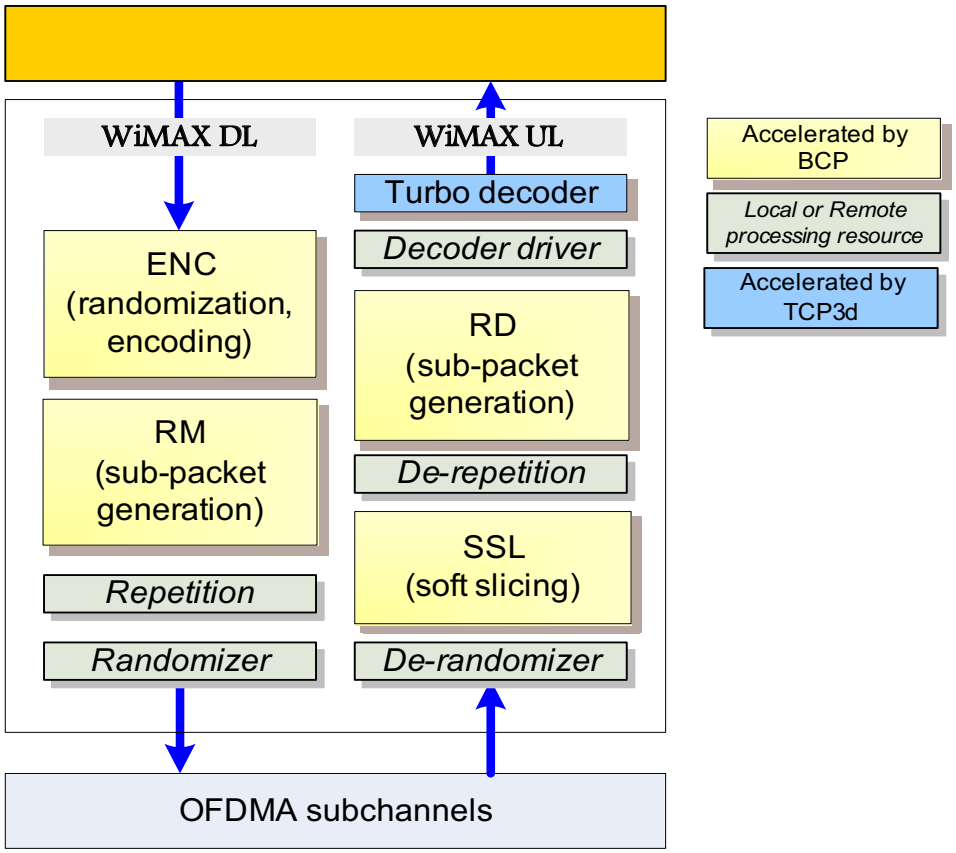
2.3.7 HSUPA PIC

Figure 2-8. Example HSUPA PIC Flow Diagram



2.3.8 WiMAX

Figure 2-9. Example WiMAX Flow Diagram



Usage

This chapter describes how to set up and use the BCP from a user's point of view.

Topic **Page**

3.1 Usage Overview

Use of the BCP centers around the Multicore Navigator. Please review the Multicore Navigator for KeyStone Devices User Guide to understand the key concepts of this data transport system before continuing.

After initialization, jobs are sent to the BCP in the form of packets by placing them on one of dedicated BCP transmit (TX) hardware queues. This signals the Packet DMA in the BCP Navigator interface to read them into the BCP and process them. At the output, the Packet DMA writes results back to the data buffers specified by the retrieved receive (RX) descriptor, before placing that descriptor on the proper RX queue. The RX free descriptor queue used to obtain the RX descriptor and the destination RX queue are determined from the Packet DMA RX flow table using the flow ID specified in the BCP packet.

The BCP is configured both through static memory mapped registers and dynamically through a per-packet header. Static registers are used for init-time configuration. For example, setting the queue priorities and configuring the flow table are done at init-time. Any configuration values that can change for each incoming packet are in the per-packet BCP header.

The BCP header is part of the payload of each packet and is parsed by the BCP to indicate information about the configuration of each submodule and the route through the BCP. The configuration information contained with each packet will have relevant information for the data contained within the packet.

Generally, submodules within the BCP output one packet for every one input packet. The submodules start outputting data as soon as they can without waiting for the entire packet to be completed to reduce latency. There are exceptions to the one-to-one relationship between input and output packets. In the uplink rate dematching output, every code block will be output as a separate packet to reduce latency. This feature is referred to as “one-to-many.” For uplink re-encoding, code blocks are input but the output is a transport block. This is referred to as “many-to-one.”

3.1.1 Initialization

The BCP must be initialized before use by configuring the memory mapped registers throughout the block. The register descriptions are given per submodule later in the document.

3.1.2 BCP Packet Format

BCP packets contain the data and dynamic configuration needed for processing that data. The payload of the packet sent is the data required for the first operation in the BCP path. After a submodule performs its operation, it generates a new packet where the payload is the result from that operation. The packet is then used as the input to the next operation. Header information dictates which BCP submodules are visited by the packet and the order of visitation (as well as configuration data for each submodule visited). The header and payload which comprise the BCP packet have the following characteristics.

BCP Header

- The total length of the header must be a multiple of 128 bits; the DSP must add dummy words to the end of the complete header to make it come out to a multiple of 128 bits if necessary.
- The global portion of the header is the first 64 bits of any packet. It contains 32 bits of control information followed by a 32-bit user-defined tag used to identify the packet in the data logger. The global header has information needed by all submodules such as the pointer to the beginning of the payload and radio standard. The bit format of the global header is given in Section 4.2.2.1.

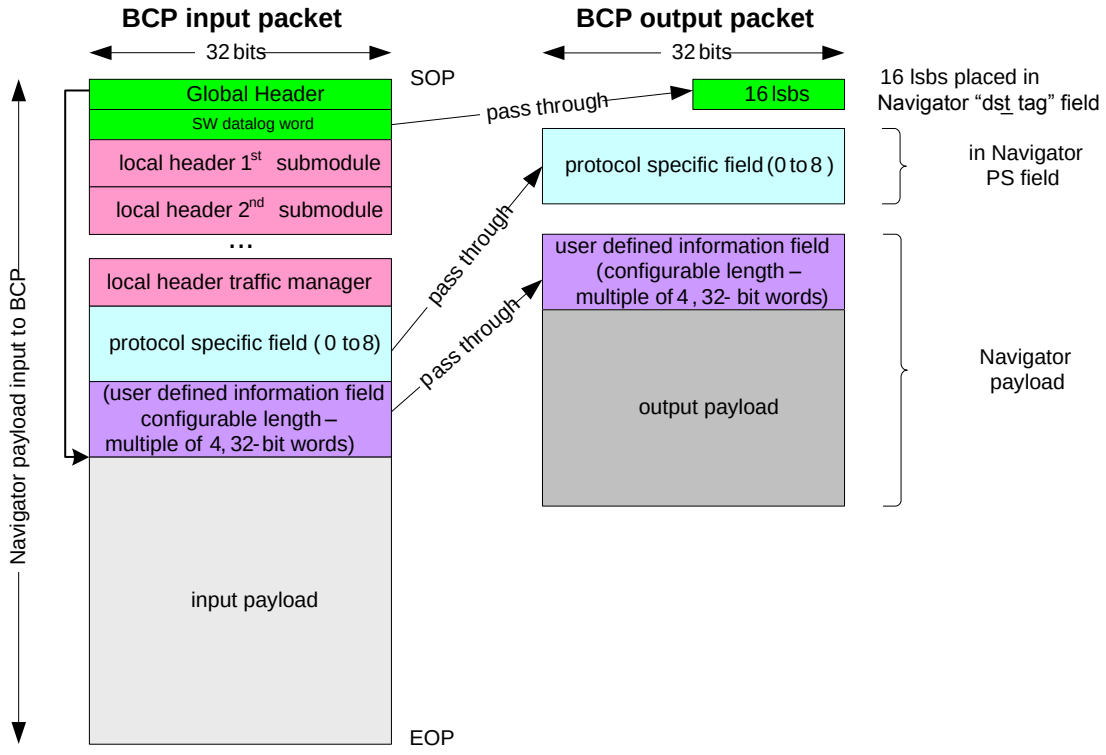
- Submodule-specific local headers have the following attributes.
  - They must each be a multiple of 32 bits.
  - They contain configuration data for each submodule.
  - They contain the length of configuration data (used to calculate beginning of next submodule header segment).
  - The submodule header lengths may depend on the mode or other bits in each submodule header.
  - The order of submodule headers dictates the order that the packet is routed inside the BCP.
  - There may be 32-bit words at the end of a local header that are not required for all scenarios. These words may be omitted by the DSP when not needed and the corresponding BCP submodules will automatically pad the missing words with zeros when using the header.
- Optionally, the traffic manager local header can include a protocol-specific (PS) info field and/or a user-defined info field that get passed-through to the output of the BCP.

BCP Payload

- Payload data in the exact format needed by the first submodule to be visited.

One important note is that the BCP header and payload are all contained in the Navigator packet payload. The BCP input does not use any of the protocol-specific, extended packet info, or other special descriptor features of the Navigator packet format. The BCP output, however, can include an optional protocol-specific field for configuring a peripheral such as SRIO and an optional user-defined information field. Also, the 16 least-significant-bits of the SW datalog word field in the BCP global header are written to the "dst_tag" Navigator descriptor field in the output packet. This is shown in Figure 3-1. In the "one-to-many" rate dematching output, all output packets have the same format – the SW datalog word, protocol-specific, and user defined information fields are duplicated for each output packet.

Figure 3-1. BCP Packet Format and Pass-Through Fields

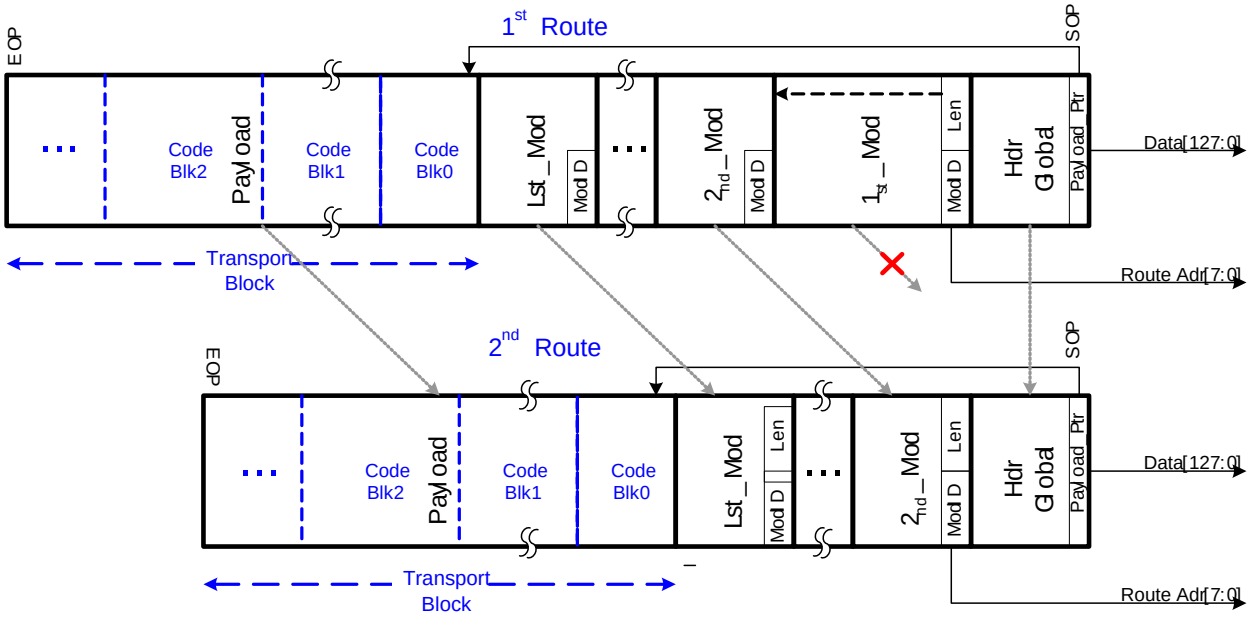


The optional protocol-specific field is part of the traffic manager local header on input, but is written to the Navigator protocol-specific field on output. The protocol-specific field can be up to 8 words (word = 32 bits) in length. Padding must be added to the end of the protocol-specific data if needed to make it a multiple of 4 32-bit words. The length of this field is set in the traffic manager local configuration header.

The optional user-defined information field in the traffic manager local header is specific to the BCP and is not part of any of the Navigator-defined fields. This field is simply passed through to the output packet, placing the data in the beginning of the data buffer of the output packet. The length of this field is specified in 128-bit quad-words and thus must be a multiple of 128-bits. It can be up to 31 128-bit quad-words. It can be used by the application for any purpose. The length of this field is set in the traffic manager local configuration header.

After each submodule is visited, the local header belonging to that submodule is removed, as shown in Figure 3-2. This way, when each submodule receives the packet, it will see its header as the first local header in the packet. The global header is also updated with a new payload pointer to be passed on to the next submodule since the total header was reduced by the length of the removed local header. This is illustrated in Figure 3-2.

Figure 3-2. Local Packet Header Removal by Submodule



The contents of the packet payload data change depending on what operation the BCP is performing. Commonly, several of the submodules operate on code blocks (similar concept for other radio standards). It is the responsibility of each submodule to partition the input data into its required granularity and to rebuild the export packet back up to the packet level. Submodule specific header information controls the segmentation operation.

The maximum total BCP packet size is only limited by the maximum payload size of the Navigator interface (different for monolithic and host packets). However, the BCP header does have a maximum allowed header size. The maximum BCP header size varies depending on which submodule the packet is entering. Table 3-1 shows the total number of 32-bit words in each submodule's BCP header FIFO. A BCP header cannot be larger than a submodule's header FIFO as the packet enters that submodule. The header size includes any protocol-specific or user-defined info, if present. Also, each FIFO is designed to be able to hold multiple headers so the submodule can potentially start processing the next packet before the previous packet header has been sent to the next submodule.

Table 3-1. BCP Header FIFO Sizes (in 32-Bit Words)

Submodule	BCP Header FIFO Size (32-bit words)
RM	496
RD	496
ENC	496
MOD	240
COR	240
INT	240
DNT	240
SSL	240
CRC	240
DIO	240

One other note is that there are two possible ways the BCP header and payload data can be partitioned using Navigator descriptors. The first is to place the BCP header and payload data in one contiguous data buffer pointed to by one descriptor. Either a monolithic or host descriptor may be used. The second method is to place the BCP header and payload data in different data buffers, each pointed to by a host descriptor where the first descriptor is linked to the second. This allows the BCP header to be created in advance if desired, and then just linked to the payload data descriptor when the data becomes available.

3.2 BCP Packet Behavior

3.2.1 Flow Tables

Every incoming packet contains, in the BCP global header, a flow identification (flowID). The BCP uses the flowID to index the flow table in the Packet DMA to determine which free descriptor queue and RX destination queue (among other things) to use for the outgoing packet when processing is completed. However, there is also a corresponding flow table in the traffic manager that holds additional information indexed by flowID that will be applied to the packet. This table holds information like output QFIFO, endian formatting, and DSP core interrupt enabling. Both tables have 64 entries and are programmed through registers at initialization time. The Packet DMA flow table is described in the Navigator user's guide. The traffic manager flow table is described in the traffic manager memory mapped register section.

3.2.2 Routing

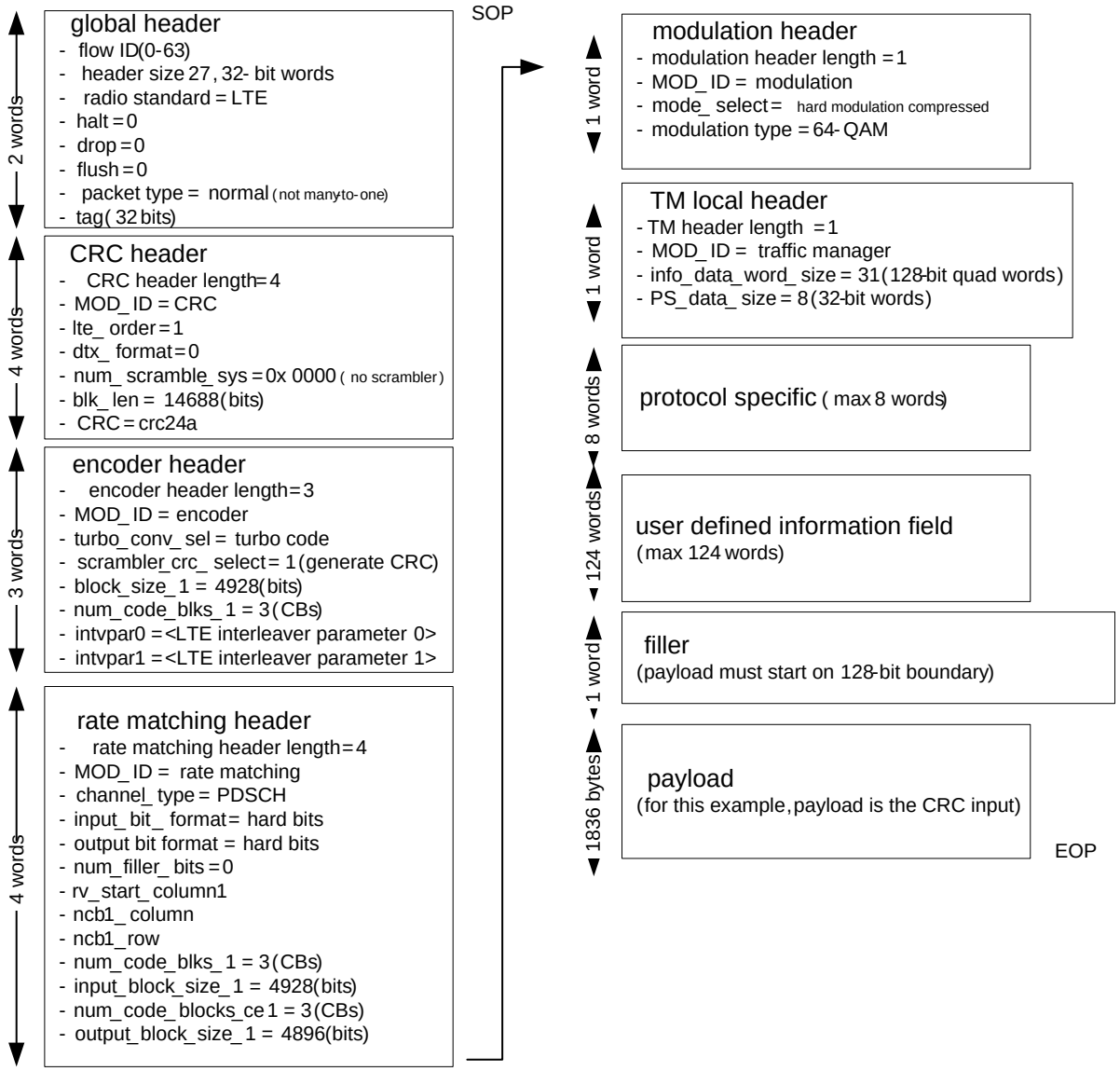
The MOD_ID (submodule ID) fields in the submodule local headers in a BCP packet control routing. The MOD_ID in each local header is the MOD_ID corresponding to the targeted submodule. For example, a local header for the encoder submodule has a MOD_ID that corresponds to the encoder submodule.

As the packet enters the BCP, the traffic manager reads the first MOD_ID to determine where to send the packet first. Each submodule routes the packet to the subsequent submodule specified by the MOD_ID in the next submodule header. The traffic manager also has a local header and a MOD_ID. When the MOD_ID is for the traffic manager, the packet is done being routed through the BCP and returns to the traffic manager to be output by the Packet DMA.

NOTE: Note that a packet can be sent directly to the traffic manager without going through any other submodules. This allows for basic loopback testing, where the data payload is not modified.

Figure 3-3 shows an example of a BCP packet formed for the LTE downlink.

Figure 3-3. Example Packet for LTE Downlink



In this case, the input is sent to the CRC submodule followed by the ENC, RM, and MOD submodules, before returning to the traffic manager. The maximum allowed number of user-defined information and protocol-specific pass-through fields were added to illustrate what is possible (these fields are optional).

In this example, the header length is 2+4+3+4+1+1+124+8+1 = 148, 32-bit words or 592 bytes for the header + 1836 bytes for the payload.

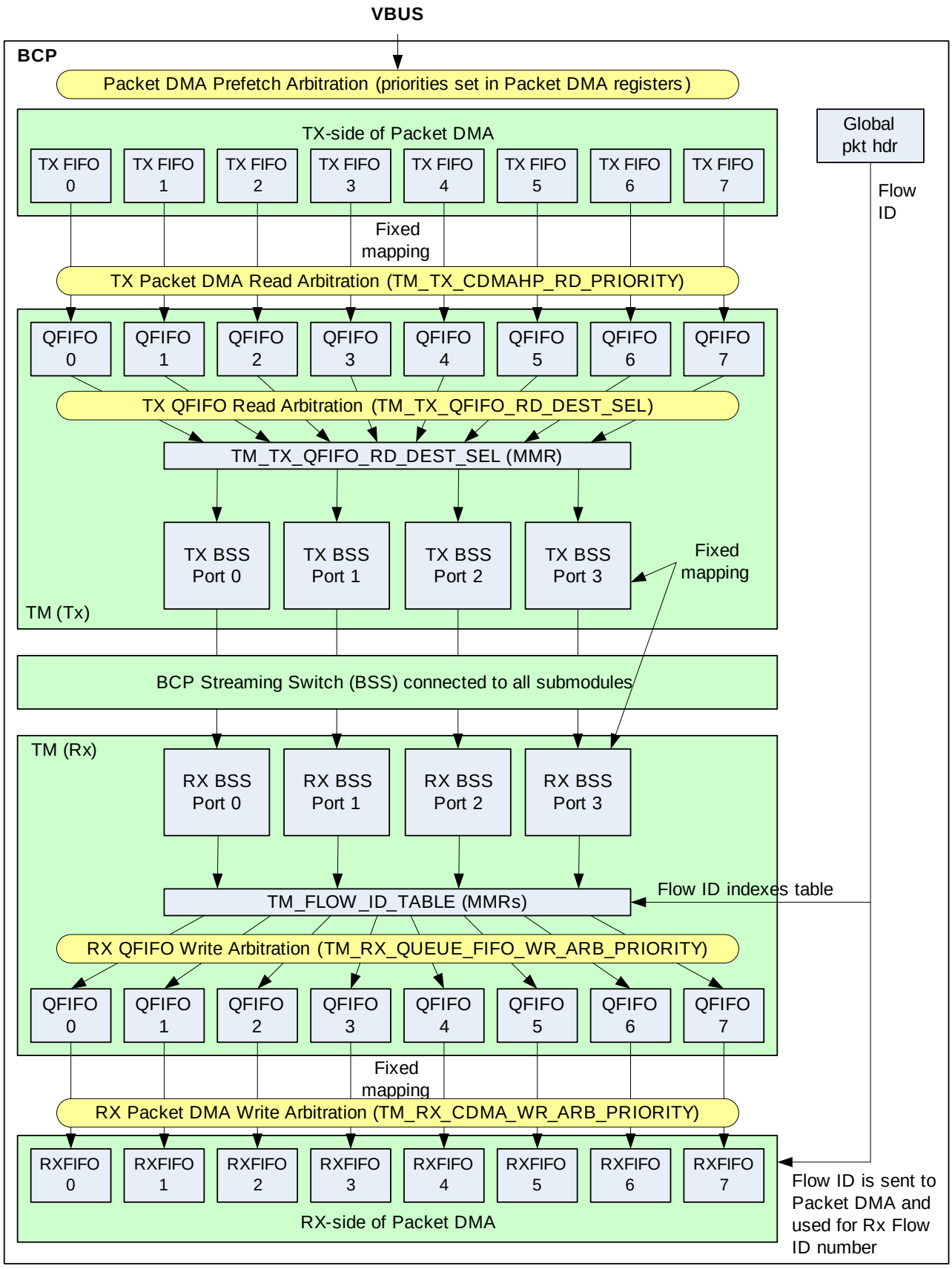
3.2.3 Queue Mapping and Prioritization

The BCP has eight ingress transmit queues corresponding to the eight transmit channels in the BCP Packet DMA. There are also eight egress receive channels. This allows up to eight simultaneous packets to be in-flight for ingress and another eight for egress. Each transmit queue is mapped to the corresponding QFIFO buffer inside the traffic manager (TM) which is then mapped via a TM configuration register (setup during initialization) to one of the four ports that connect the TM to the BCP streaming switch (BSS). Simultaneous traffic is supported on all four ports of the BSS. For example, uplink, downlink and PUCCH traffic can all flow simultaneously without blocking.

Packets always exit and enter the TM on the same BSS port. A packet that entered the BSS on port n will also exit the BSS on port n. But on output, there is no direct mapping between the port number and QFIFO. Instead, the QFIFO, RX free descriptor queue and RX destination queue are all selected according to the flow ID in the BCP packet header.

The queue mapping and prioritization is shown in Figure 3-4. Ingress data enters from the top and egress data exits at the bottom.

Figure 3-4. BCP Queue Mapping and Prioritization



There are three levels of arbitration in the transmit (ingress) path as shown in the yellow tasks in the previous figure. The first arbitration is done within the Packet DMA. The Packet DMA has eight FIFOs, one for each input transmit queue. The Packet DMA will pre-fetch data from each queue when there is a packet pending and there is room in the FIFO according to a simple, configurable priority scheme. Each queue is programmed with one of four priority levels. Pre-fetching is round robin within a priority level and drops to the next priority level when the FIFOs are filled or nothing is pending on the higher level. Arbitration is on a fetch packet granularity of 64 bytes. The TM has no control over this pre-fetching. When Packet DMA FIFOs have at least one, 128-bit word (or an EOP), signals indicate to the TM that data is available on those queues.

Inside the TM, there are two more levels of transmit path arbitration. The TX Packet DMA read arbitration level is used to fill an additional eight QFIFOs inside the TM – again one for each of the eight transmit queues. The eight queues are each assigned one of four priority levels and the same type of priority scheduling is done to fill the QFIFOs pulling data from the Packet DMA when there is data available and room in the QFIFOs. Arbitration is done on the granularity of 128 bits, meaning that every cycle a new queue can be selected since the bus is 128 bits wide. When any of the QFIFOs has at least 1, 128-bit word (or an EOP) it notifies the last level of arbitration that there is data available.

The final level, TX QFIFO read arbitration selects one of the eight TX QFIFOs for sending to one of the four ports on the BCP streaming switch (BSS). A programmable register in the BCP (tx_queue_dest_sel) assigns each of the eight QFIFOs to one of the four ports. All queues that are mapped to the same port form a group. Within each group, arbitration is again done using a simple priority scheme as described before with up to four levels of priorities. At this level, arbitration is done on the granularity of a BCP packet (which could be more than one Navigator packet if the “many-to-one” feature is used). In other words, a BSS port will complete one packet before switching to another packet.

When the TX QFIFO read arbitration selects a queue, the TX Packet DMA read arbitration priority level for the selected queue is raised by four priority levels. The original priority levels are 0, 1, 2, and 3 with 0 being the highest. If a queue has TX Packet DMA read arbitration priority n, the new priority level will be n – 4 effectively creating eight priority levels with –4 being the highest. Once the end of a packet is reached at the QFIFO read level, the priority returns to the original level. The TX QFIFO read priority and the Packet DMA pre-fetch priority are not affected. This priority raising feature can be disabled.

The TM is also responsible for controlling/regulating outbound traffic. The arbitration scheme implemented for outbound traffic is identical to the one used for incoming traffic except that there is no priority raising feature for the outbound traffic. As with incoming traffic, it is possible that multiple packets coming from different ports on the BCP BSS switch are destined to one outgoing QFIFO. Even though these packets might all be going to different flows (different free/destination queues), there is still contention for the QFIFO in this case. Priority levels can be set independently for incoming and outgoing traffic. This allows, for example, priority scheduling of the input and round-robin scheduling of the output.

The QFIFOs are the same size for incoming and outbound traffic but Packet DMA FIFOs are slightly larger for outbound traffic. The Packet DMA FIFOs are 640 bytes for the outgoing packets as opposed to 512 bytes for inbound traffic as shown in Figure 3-4.

3.2.4 Special Features

The BCP supports some special features designed for debugging and synchronization.

3.2.4.1 Flush

With the synchronization of transport channel encoding and radio frame segmentation in mind, the BCP provides a flush feature. A flush is indicated by a bit in the global packet header whose purpose is to flush the pipeline in the BCP. Like all other packets, a flush packet is configured with a path through the BCP. The flush packet exits the BCP through the Packet DMA and is placed on a receive queue like all other packets.

When a packet with flush indicated (a flush packet) enters the BCP on a given traffic manager QFIFO (from the corresponding transmit queue), no other packets are accepted on that QFIFO until the flushed packet exits the BCP and is written to the RX destination queue. Packets coming in using other QFIFOs (transmit queues) can continue to enter the BCP.

The Packet DMA will prefetch data even when the traffic manager stops a packet from entering the BCP. FIFOs inside the Packet DMA hold the data. Therefore, a flush packet cannot prevent the Packet DMA from starting to read the next packet into its FIFOs; it can only stop those packets from entering the traffic manager. This is why the drop feature was also added to support synchronization.

3.2.4.2 Drop

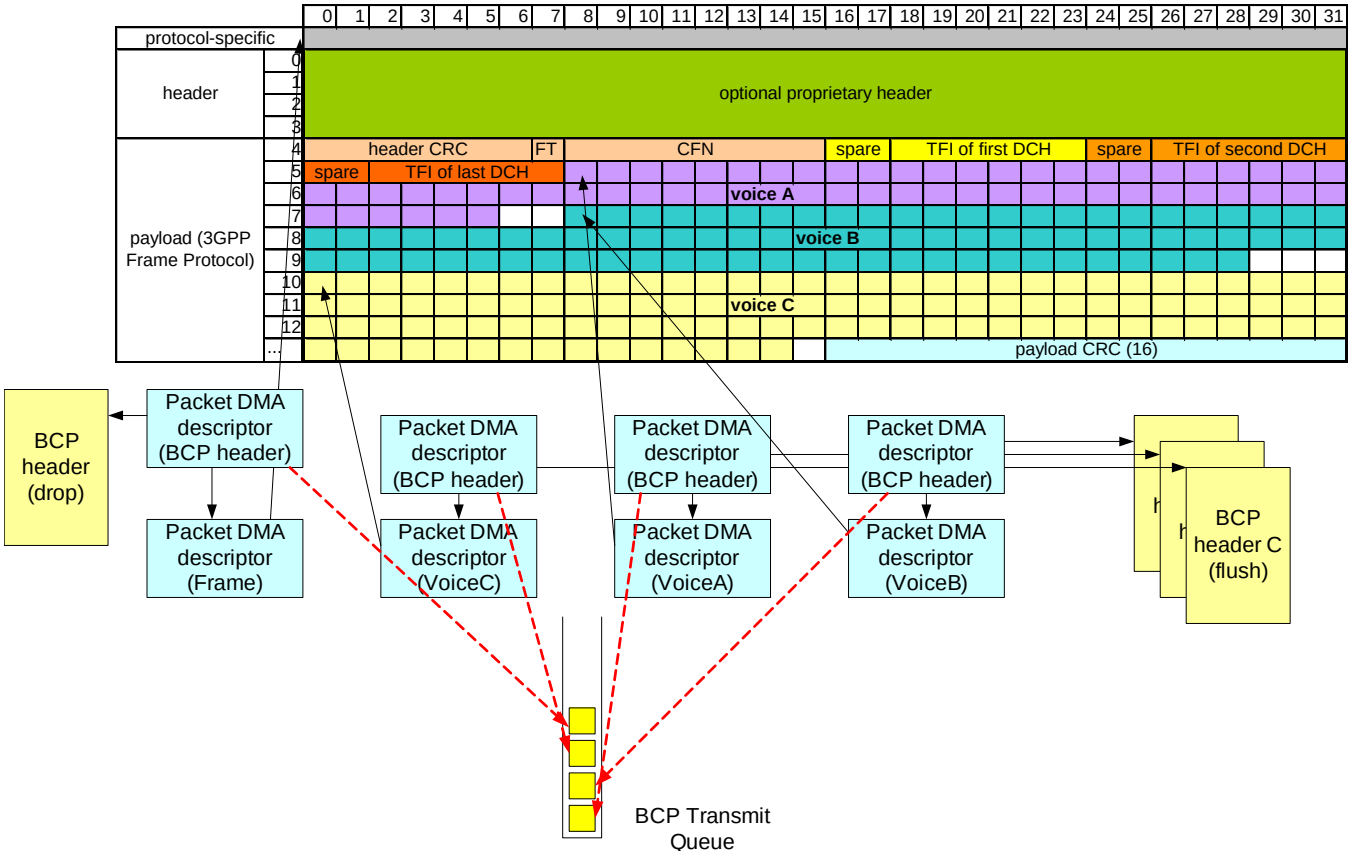
A drop is indicated by a bit in the global packet header. A packet with drop indicated is read by the Packet DMA but the contents are ignored by the BCP. A dropped packet is not sent to any of the BCP submodules and the BCP generates no output from a dropped packet. However, from the point of view of the Packet DMA, it is a normal Navigator packet.

The intent of drop is to work together with flush to enforce synchronization. If there are three packets written to the same BCP transmit queue, packets A, B, and C, drop can be used to ensure that packet C is not consumed by the BCP until packet A is complete and written to memory. This could be useful if packet C uses the results of packet A. In this case, packet A would be tagged with flush, packet B is a dummy packet tagged with drop, and packet C is a normal packet. There are no special limits on the size of a drop packet. However, in the example, packet B must be long enough to fill the Packet DMA FIFOs to prevent the Packet DMA from starting to read packet C.

3.2.4.2.1 Example Using Flush and Drop

Here is an example usage of the flush and drop features. In this example, a 3GPP frame is received from the MAC. An optional proprietary header could be appended to the beginning of the frame. There could also be a Navigator protocol-specific field. This is illustrated in Figure 3-5.

Figure 3-5. Flush and Drop Example



In this example, one frame contains three downlink transport channels: Voice A, Voice B, and Voice C. One packet is formed to point to the beginning of the frame (to keep track of the memory holding everything before the Voice A data) and three other packets pointing to Voice A, Voice B, and Voice C transport channels. Each packet is formed using two Navigator host type descriptors. The first descriptor points to the BCP header for the packet and is linked to the second descriptor which points to the payload data. It is assumed that the BCP headers were created at initialization time and stored in memory.

The Voice C packet is tagged with the “flush” indicator so the BCP will not accept another packet on the BCP transmit queue until the Voice C packet processing is complete and the results written to the destination queue. If desired, the frame descriptor can be pushed onto the BCP transmit queue tagged with “drop” to make the Packet DMA recycle it. The BCP will then read the frame packet but will ignore the contents and produce no output. Once the BCP finishes reading the frame packet, the Packet DMA will recycle the descriptor according to normal Navigator protocols. To reduce overhead, the length of the frame packet can be set to a small number of bytes – the Navigator system preserves the original length of the buffer.

3.2.4.3 Halt

If the halt bit is set in the global header, the entire BCP halts once all in-progress packets are completed (including the packet that had the halt bit set). No new packets are taken on any of the input queues. Halt is intended for debugging only. Halt can generate an interrupt once the BCP is idle. After the halt interrupt is cleared, there is a register that can be written to clear the halt and resume normal operation.

3.2.4.4 Many-To-One

For the purposes of reducing latency, it is sometimes better to start sending a transport block or transport channel into the BCP before the entire block is available. As an example, in the case of uplink re-encoding, the output of the TCP3d is sent back to the BCP for rate matching and modulation. An entire transport block which is the natural delimiter for a BCP packet can be very large. In this case, it would be better to send each decoded code block to the BCP as it is available so the BCP can start processing the data as it is available.

The “many-to-one” feature was envisioned to support uplink re-encoding but it can be used for any purpose with any BCP submodule. It allows multiple, consecutive Navigator packets to be concatenated internally within the BCP by the traffic manager and treated as a single packet. “Many-to-one” packets must be consecutive on a queue without any other packets in between. Therefore, this requires that one of the BCP transmit queues be dedicated to a user when this feature is used to prevent another user from writing to the same queue and interrupting the consecutive string of “many-to-one” packets.

Two packet mode bits are defined in the global BCP packet header to indicate that a given input Navigator packet is a normal packet, the start of a “many-to-one” packet (SOM), the middle of a “many-to-one” packet (MOM), or the end of a “many-to-one” packet (EOM). To concatenate Navigator packets into one BCP packet, the first Navigator packet is tagged with SOM, the next zero or more Navigator packets with MOM, and the last with EOM. The MOM and EOM packets do not need to include local header configuration, since the information from the SOM packet will be used for the entire concatenated packet. The MOM and EOM packets do need a global header - any other header, if present, is ignored by the BCP.

SOM and MOM packets must have a multiple of 16 bytes. If an SOM or MOM packet does not have a multiple of 16 bytes, pad bytes will be added by the BCP to make it a multiple of 16 bytes and an error will be flagged.

Once a BCP packet starts entering the BCP, be it a “many-to-one” packet or a normal packet, no other packets can enter on the same input queue or go to the same BCP submodule until the first packet is consumed. The BCP submodules are single-threaded, meaning that they can only process one packet at a time. Therefore, if a “many-to-one” packet uses a given submodule, that submodule will be busy until the end of the packet. Even if there is a long pause between the SOM and EOM packets, the submodules involved will be busy throughout the duration of the packet.

Unlike a normal Navigator packet that is guaranteed to have a start and end of packet (SOP and EOP), programming errors could cause an out of order or unexpected SOM, MOM, or EOM. If an MOM or EOM is received without having first received an SOM, an error is flagged and the Navigator packet is dropped (no BCP output). All MOM and EOM packets will be dropped until the BCP receives either a normal or SOM packet.

After an SOM and zero or more MOM packets are received, an error occurs if a normal or SOM packet is received. In this case, if a normal packet is received, it is considered like an EOM packet and an error is flagged. If an SOM packet is received, it is treated like an MOM packet and an error is flagged. The BCP will continue normally after the errored packets are completed.

If an SOM and zero or more MOM packets are received and an EOM packet is missed, the affected input queue on the BCP will stall until another Navigator packet on the affected queue is received. A normal packet that does nothing (a dummy packet) could be used to flush a thread if there is a concern about hanging the BCP.

3.2.4.5 One-To-Many

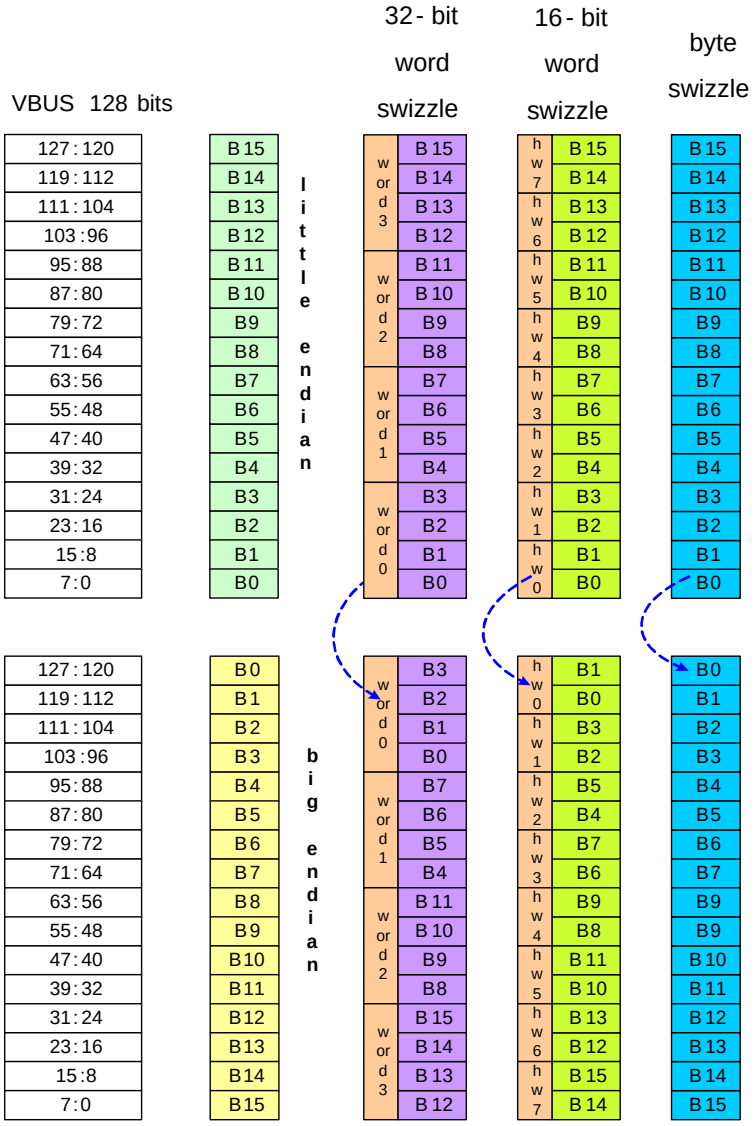
For the purposes of reducing latency, it is better to start processing the output of the rate de-matching submodule before the entire transport block is processed. Unlike the “many-to-one” feature which can be applied to any BCP submodule, this “one-to-many” feature is only used to output one code block per packet out of the rate de-matching submodule. The “one-to-many” feature is not a programmable option – the output of rate dematching is always one code block per packet.

3.3 Data Reformatting and Endian Conversion

The selection between big and little-endian is set according to the device settings through an input signal to the BCP. However, inside the BCP all submodules have a fixed, little endian format. Therefore, inbound and outbound traffic can be reformatted as needed. This reformatting is done within the traffic manager and direct I/O modules. The effect of endian settings on the payload and payload reformatting is configured in the BCP flow table on a per-flow basis.

The BCP header and payload are treated separately. The header is always considered as 32-bit words. The payload can be considered as 8, 16, or 32-bit units as specified in the flow table. When the device is set to little-endian, there is no endian correction. When it is set to big-endian, the format conversion is done as shown in Figure 3-6.

Figure 3-6. Endian Conversion



In addition to endian conversion, the BCP can also reformat the data by swapping pairs of 8-bit, 16-bit, or 32-bit words and swapping bits within an 8, 16, or 32-bit word. Format conversion is done after endian conversion for inbound traffic and before endian conversion for outbound traffic. All endian and format conversion can be set separately for input and output data.

The following tables show the formatting changes available for 8, 16 and 32-bit words on a 128 bit input word. They are either swapping words on 8, 16 or 32-bit boundaries or bit reversals within 8, 16 or 32-bit boundaries.

Table 3-2. 8-Bit Byte Swap

INPUT	127-120	119-112	111-104	103-96	95-88	87-80	79-72	71-64	63-56	55-48	47-40	39-32	31-24	23-16	15-8	7-0
OUTPUT	119-112	127-120	103-96	111-104	87-80	95-88	71-64	79-72	55-48	63-56	39-32	47-40	23-16	31-24	7-0	15-8

Table 3-3. 16-Bit Swap

INPUT	127-112	111-96	95-80	79-64	63-48	47-32	31-16	15-0
OUTPUT	111-96	127-112	79-64	95-80	47-32	63-48	15-0	31-16

Table 3-4. 32-Bit Word Swap

INPUT	127-96	95-64	63-32	31-0
OUTPUT	95-64	127-96	31-0	63-32

Table 3-5. 8-Bit Bit Reversal

INPUT	127-120	119-112	111-104	103-96	95-88	87-80	79-72	71-64	63-56	55-48	47-40	39-32	31-24	23-16	15-8	7-0
OUTPUT	120-127	112-119	104-111	96-103	88-95	80-87	72-79	64-71	56-63	48-55	40-47	32-39	24-31	16-23	8-15	0-7

Table 3-6. 16-Bit Bit Reversal

INPUT	127-112	111-96	95-80	79-64	63-48	47-32	31-16	15-0
OUTPUT	112-127	96-111	80-95	64-79	48-63	32-47	16-31	0-15

Table 3-7. 32-Bit Bit Reversal

INPUT	127-96	95-64	63-32	31-0
OUTPUT	96-127	64-95	32-63	0-31

3.4 Debug

There are several debug capabilities built into the BCP. The normal capability of the BCP to route a packet through any subset of submodules offers the possibility of sending data to only one submodule at a time. In addition, every submodule detects and reports errors. These errors can trigger interrupts or be logged by the data logger inside the submodule. The BCP also supports emulation suspend.

3.4.1 Emulation Suspend

In response to an emulation suspend signal, the BCP will stop taking incoming data and will stop sending outgoing data. When the emulation signal is removed, data flow will continue. Only “soft” emulation suspend is supported by the BCP, meaning that if there are incoming or outgoing packets in process, they will continue until the end of the packets.

3.4.2 Error Reporting

Every submodule on the BCP can report up to seven error conditions. The definition of these errors is submodule-specific. These errors are generally caused by invalid parameter settings, a packet length that does not match the parameter settings, routing to invalid submodules, or an event from the data logger which will be explained shortly.

An error can cause a maskable interrupt to any combination of the DSP cores. For every submodule, there are four mask registers intended for the four DSP cores. Each DSP can mask any combination of errors.

In addition, interrupts for any DSP can be enabled or disabled based on the flow_ID of the packet that caused the error. There is an entry for all flow_IDs to enable or disable any combination of DSP cores. For example, if a packet with a flow_ID = 9 causes an error that was not masked and the DSP interrupt selection for flow_ID 9 is ‘0011’b, then DSP cores 0 and 1 will be interrupted.

A data logger event is not an error caused by a packet and therefore it is not affected by the flow ID but it can still be masked.

The BCP streaming interface contains a four-bit error signal, one bit per DSP core. A submodule sets these four bits according to whether there are any unmasked errors for each DSP. These bits are valid at the end of the packet. By the time the packet has been sent through all submodules, the four bits are the logical OR from all submodules. These four bits are sent using the Navigator error flag bits and will be available in the Navigator descriptor. When using the BCP remotely over SRIO for example, these bits will not be available because they are not transmitted over SRIO.

Every submodule contains a memory mapped register that contains the packet tag of the packet currently being output. It can be read by any DSP core.

Four interrupt signals are sent to the top level error handling aggregator from each submodule. A signal is sent by a submodule if an error bit is set and the error for that DSP core was not masked and that DSP core error interrupt is enabled in the flow table. If the interrupt is set by any of the submodules, an interrupt will be generated.

Through a top-level register (located in the traffic manager register space), an interrupt for any DSP core can also cause the BCP to halt. A BCP halt is a global halt and it means that the BCP will not accept any more input packets on any threads once the current input packets have been completely read. This bit can also be set manually by the DSP to generate a BCP halt. Clearing the halt bit allows the BCP to continue. If the halt bit is cleared but an interrupt event is still pending and the halt is still enabled, the BCP will immediately halt again. A top-level status register will indicate that the BCP is halted.

In the traffic manager there is a top-level status register for each of the four DSP cores. Each status register contains one bit per submodule that indicates if there has been an error event from that submodule to the DSP core.

3.4.3 Data Logger

The event data logger is a length-256 circular buffer built in to every submodule. It logs packets that exit the submodule. The data logger logs:

- a SW datalog word that is part of the global header in each packet – the word is filled in by the DSP
- 32-bit BCP top-level register value typically used for radio timestamps; the register can be written periodically (for example, every 70µs in LTE)
- a locally generated 24-bit free running hardware timestamp
- any errors that were detected in the packet

The data logger has several different run modes. It can capture:

- only packets with errors
- all packets
- all packets starting on the next error
- all packets until there is an error and then stop
- In mode 1, the data logger allows address wrapping.

- In mode 2, the data logger can be configured to allow address wrapping or to stop once it reaches the end of the memory.
- In mode 3, capturing will stop once the end of the memory is reached.
- In the last mode, capturing is continuous with wrapping until an error occurs.

In all cases, when the data logger stops for any of the reasons listed above, it generates a data logger event which, like the other errors, can cause an interrupt if it is not masked. The data logger data can be read through memory mapped registers.

The status of each data logger can be read. Status bits include:

- running – this bit indicates whether the data logger has stopped or not
- has_error – this bit indicates that there is at least one error in the data logger
- wrap_count – a counter from 0 to 255 that tells the number of times the logger has wrapped (stops at 255)
- write pointer: the current write pointer into the data logger memory

There is a top-level BCP register (in the traffic manager register space) with one bit per submodule that allows the DSP to clear any or all submodule data loggers. There is also a corresponding register with one bit per submodule that will cause the data logger to be in the hold state (like a pause). Clearing this bit allows the data logger to continue but any packets that passed while the data logger is in hold would be missed. Although the BCP is a shared resource, the data logger is global and should only be enabled or cleared by one master. When the data logger is cleared, the error count returns to zero so that “has_error” indication is cleared, the data logger returns to a running state in whatever mode is programmed and the wrap_count returns to zero. The write pointer is reset.

When a DSP is interrupted, it must clear the interrupt before it will receive another interrupt. If the DSP clears an interrupt and the submodule that generated that interrupt was not cleared, the DSP will immediately receive another interrupt.

Common programming errors will be detected. However, there is no guarantee that all errors will be detected (such as combinations of values that are not legal). The goal is that no errors will lock up a BCP submodule.

The data logger control and interrupt registers are instantiated in each submodule at the same address offset relative to the submodule’s base address. The register definitions are given in the next two sections and apply to all BCP submodules (except the Packet DMA, which does not have a data logger).

3.4.3.1 Data Logger Control Registers

The data logger control registers start at an address offset of 0xF0 in each submodule and appear in the same order, as described below.

Table 3-8. Data Logger Control Register Address Offsets (relative to each submodule base address)

DLG Control Registers Memory Map		
Register	**Address**	**Description**
DATA_LOGGER_CTL	0xF0	Data logger control Register
DATA_LOGGER_STATUS	0xF4	Datalogger Status Register
GLOBAL_HDR (SW_datalog_word)	0xF8	SW datalog word from current global header

Figure 3-7. DATA_LOGGER_CTL Register - Address [0xF0]

31 25	24 16	15 3	2 0
Reserved	dlg_err_mask	Reserved	dlg_ctl
RW - 0x0	RW - 0x0	RW - 0x0	RW - 0x0

Legend: R = Read only; W = Write only

Table 3-9. DATA_LOGGER_CTL Register Details

Bits	Field Name	Description
31-25	Reserved	Reserved.
24-16	dlg_err_mask	Disables the corresponding error source when a bit is set.
15-3	Reserved	Reserved.
2-0	dlg_ctl	Selects the mode that DLG is running. The modes are listed in a table later in this section.

Figure 3-8. DATA_LOGGER_STATUS Register — Address [0xF4]

31 18	17	16	15 8	7 0
Reserved	dlg_running	dlg_mem_has_error	dlg_wr_wrap	dlg_wr_ptr
R - 0x0	R - 0x0	R - 0x0	R - 0x0	R - 0x0

Legend: R = Read only

Table 3-10. DATA_LOGGER_STATUS Register Details

Bits	Field Name	Description
31-18	Reserved	Reserved
17	dlg_running	Set while capturing data. Clears when the last location of the memory is written if the selected mode stops capturing when memory is full.
16	dlg_mem_has_error	Indicates there is at least one error currently stored in the memory.
15-8	dlg_wr_wrap	Number of times writing to memory wrapped to 0. Stops at 0xFF.
7-0	dlg_wr_ptr	The current value of the DLG RAM write pointer.

Figure 3-9. GLOBAL_HDR Register (SW datalog word) — Address [0xF8]

31 0
global_hdr (SW datalog word)
R - 0x0

Legend: R = Read only

Table 3-11. GLOBAL_HDR Register Details

Bits	Field Name	Description
31-0	global_hdr (SW datalog word)	SW datalog word from current global header. This is the second 32-bit field in the packet's global header that is loaded into the DLG RAM after this submodule sends the output data.

Changing the mode (Table 3-12) should always go from DLG_IDLE to the selected data capture setting and then back to DLG_IDLE or DLG_HOLD followed by DLG_IDLE.

Table 3-12. DLG CTL Mode Settings

Bits	Field Name	Description
0	DLG_IDLE	Data logger does not capture any information and write pointer is reset to address 0.
1	DLG_ERRORS_ONLY	Stores information on all errored packets. Write counter is allowed to wrap so writes continue indefinitely.
2	DLG_START_ON_ERROR	Starts capturing data on the next error and captures information for all the following packets until all memory locations have been written. Write address starts at 0.
3	DLG_STOP_ON_ERROR	Stores data from all packets until an error occurs. When the error occurs it stores that packets data and stops collecting data.
4	DLG_FREE_RUN	Stores data from all packets. Write counter is allowed to wrap so writes continue indefinitely
5	DLG_ONESHOT	Stores data from all packets starting at location 0 and continues until all memory has been written and then stops.
6	Reserved	Reserved
7	DLG_HOLD	Does not store any more information but does not clear the write pointer.

3.4.3.2 Data Logger Interrupt Registers

The data logger interrupt registers (Table 3-13) start at an address offset of 0x80 in each submodule and appear in the same order.

Table 3-13. Data Logger Interrupt Register Address Offsets (relative to each submodule base address)

DLG Interrupt Registers Memory Map		
Register	**Address**	**Description**
DLG Interrupt Registers for Core 0		
intr_irs_0	0x80	Interrupt Raw Status Register for Core 0
intr_irs_set_0	0x84	Interrupt Set Raw Status Register for Core 0
intr_irs_clr_0	0x88	Interrupt Clear Raw Status Register for Core 0
intr_en_0	0x8C	Enable Interrupt Register for Core 0
intr_en_set_0	0x90	Set Enable Interrupt Register for Core 0
intr_en_clr_0	0x94	Clear Enable Interrupt Register for Core 0
intr_en_sts_0	0x98	Enabled Interrupt Register for Core 0
DLG Interrupt Registers for Core 1		
intr_irs_1	0x9C	Interrupt Raw Status Register for Core 1
intr_irs_set_1	0xA0	Interrupt Set Raw Status Register for Core 1
intr_irs_clr_1	0xA4	Interrupt Clear Raw Status Register for Core 1
intr_en_1	0xA8	Enable Interrupt Register for Core 1
intr_en_set_1	0xAC	Set Enable Interrupt Register for Core 1
intr_en_clr_1	0xB0	Clear Enable Interrupt Register for Core 1
intr_en_sts_1	0xB4	Enabled Interrupt Register for Core 1
DLG Interrupt Registers for Core 2		
intr_irs_2	0xB8	Interrupt Raw Status Register for Core 2
intr_irs_set_2	0xBC	Interrupt Set Raw Status Register for Core 2
intr_irs_clr_2	0xC0	Interrupt Clear Raw Status Register for Core 2
intr_en_2	0xC4	Enable Interrupt Register for Core 2
intr_en_set_2	0xC8	Set Enable Interrupt Register for Core 2
intr_en_clr_2	0xCC	Clear Enable Interrupt Register for Core 2
intr_en_sts_2	0xD0	Enabled Interrupt Register for Core 2

Table 3-13. Data Logger Interrupt Register Address Offsets (relative to each submodule base address) (continued)

DLG Interrupt Registers Memory Map		
Register	**Address**	**Description**
DLG Interrupt Registers for Core 3		
intr_irs_3	0xD4	Interrupt Raw Status Register for Core 3
intr_irs_set_3	0xD8	Interrupt Set Raw Status Register for Core 3
intr_irs_clr_3	0xDC	Interrupt Clear Raw Status Register for Core 3
intr_en_3	0xE0	Enable Interrupt Register for Core 3
intr_en_set_3	0xE4	Set Enable Interrupt Register for Core 3
intr_en_clr_3	0xE8	Clear Enable Interrupt Register for Core 3
intr_en_sts_3	0xEC	Enabled Interrupt Register for Core 3

The bit fields listed in the register descriptions in the following series of tables correspond to events as follows:

- “eng_error”s are error conditions from each submodule
- “dlg_done” event indicates that data logger in this submodule has finished capturing data
- “ppb_error” indicates that the BCP streaming switch port for this submodule detected one of the following errors:
  - MOD_ID addresses either a non-existing submodule or itself
  - the payload_ptr was larger than the packet (generates an early EOP)
  - the BCP header was larger than the header FIFO in the engine

Figure 3-10. Interrupt Raw Status Register (intr_irs) — Address [0x80]

31 9	8 2	1	0
Reserved	eng_error	dlg_done_int	ppb_err
R - 0x0	R - 0x0	R - 0x0	R - 0x0

Legend: R = Read only

Table 3-14. Interrupt Raw Status Register (intr_irs) Details

Bits	Field Name	Description
31-9	Reserved	Reserved.
8-2	eng_error	Set engine Errors. Write ‘1’ to set.
1	dlg_done_int	Set data logger done has occurred. Write ‘1’ to set.
0	ppb_err	Set PPB Error. Write ‘1’ to set.

Figure 3-11. Interrupt Raw Status Set Register (intr_irs_set) — Address [0x84]

31 9	8 2	1	0
Reserved	eng_error	dlg_done_int	ppb_err
W - 0x0	W - 0x0	W - 0x0	W - 0x0

Legend: W = Write only

Table 3-15. Interrupt Raw Status Set Register (intr_irs_set) Details

Bits	Field Name	Description
31-9	Reserved	Reserved.
8-2	eng_error	Set engine Errors. Write ‘1’ to set.
1	dlg_done_int	Set data logger done has occurred. Write ‘1’ to set.
0	ppb_err	Set PPB Error. Write ‘1’ to set.

Figure 3-12. Interrupt Raw Status Clear Register (intr_irs_clr) — Address [0x88]

31 9	8 2	1	0
Reserved	eng_error	dlg_done_int	ppb_err
W - 0x0	W - 0x0	W - 0x0	W - 0x0

Legend: W = Write only

Table 3-16. Interrupt Raw Status Clear Register (intr_irs_clr) Details

Bits	Field Name	Description
31-9	Reserved	Reserved.
8-2	eng_error	Clear engine Errors. Write ‘1’ to clear.
1	dlg_done_int	Clear data logger done has occurred. Write ‘1’ to clear.
0	ppb_err	Clear PPB Error. Write ‘1’ to clear.

Figure 3-13. Interrupt Enable Register (intr_en) — Address [0x8C]

31 9	8 2	1	0
Reserved	eng_error	dlg_done_int	ppb_err
R - 0x0	R - 0x0	R - 0x0	R - 0x0

Legend: R = Read only

Table 3-17. Interrupt Enable Register (intr_en) Details

Bits	Field Name	Description
31-9	Reserved	Reserved.
8-2	eng_error	Enable for engine Errors. ‘1’ = enabled.
1	dlg_done_int	Enable for data logger done has occurred. ‘1’ = enabled.
0	ppb_err	Enable for PPB Error. ‘1’ = enabled.

Figure 3-14. Interrupt Enable Set Register (intr_en_set) — Address [0x90]

31 9	8 2	1	0
Reserved	eng_error	dlg_done_int	ppb_err
W - 0x0	W - 0x0	W - 0x0	W - 0x0

Legend: W = Write only

Table 3-18. Interrupt Enable Set Register (intr_en_set) Details

Bits	Field Name	Description
31-9	Reserved	Reserved.
8-2	eng_error	Set to enable Engine errors. Write ‘1’ to set.
1	dlg_done_int	Set to enable Data logger done has occurred. Write ‘1’ to set.
0	ppb_err	Set to enable PPB Error. Write ‘1’ to set.

Figure 3-15. Interrupt Enable Set Register (intr_en_clr) — Address [0x94]

31 9	8 2	1	0
Reserved	eng_error	dlg_done_int	ppb_err
W - 0x0	W - 0x0	W - 0x0	W - 0x0

Legend: W = Write only

Table 3-19. Interrupt Enable Clear Register (intr_en_clr) Details

Bits	Field Name	Description
31-9	Reserved	Reserved.
8-2	eng_error	Clear Enable Engine errors. Write ‘1’ to clear.
1	dlg_done_int	Clear Enable Data logger done has occurred. Write ‘1’ to clear.
0	ppb_err	Clear Enable PPB Error. Write ‘1’ to clear.

Figure 3-16. Interrupt Enable Status Register (intr_en_sts) — Address [0x98]

31 9	8 2	1	0
Reserved	eng_error	dlg_done_int	ppb_err
R - 0x0	R - 0x0	R - 0x0	R - 0x0

Legend: R = Read only

Table 3-20. Interrupt Enabled Status Register (intr_en_sts) Details

Bits	Field Name	Description
31-9	Reserved	Reserved.
8-2	eng_error	Enabled Engine Errors. ‘1’ = event occurred.
1	dlg_done_int	Enabled Data logger done has occurred. ‘1’ = event occurred.
0	ppb_err	Enabled PPB Error. ‘1’ = event occurred.

3.4.3.3 Data Logger RAM Format

The bit format of the data logger RAM is shown in Figure 3-17.

Figure 3-17. Data Logger RAM Bit Format

127 96	95 89	88	87 64	63 32	31 0
Reserved	Eng_error[6:0]	PPB_error	Tm_dlg_hw_ Timestamp [23:0]	Tm_dlg_sw_ Timestamp [31:0]	Global_hdr_field [31:0]

A description of each field is given in Table 3-21.

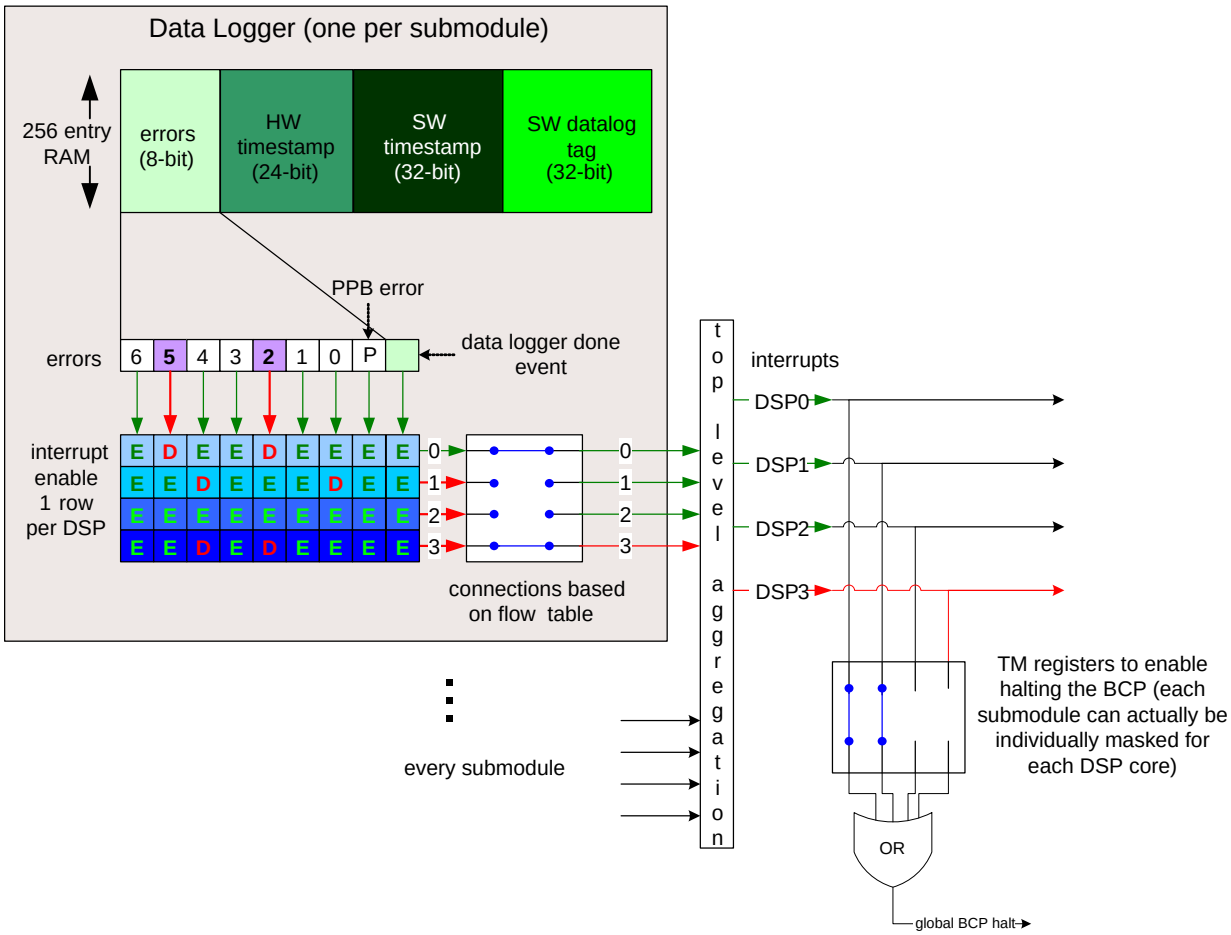
Table 3-21. Data Logger RAM Details

Bits	Field Name	Description
127-96	Reserved	This field will return 0s.
95-89	Eng_error[6:0]	Error conditions detected by the submodule.
88	PPB_error	Indicates that the BCP streaming switch port for this submodule detected one of the following errors: MOD_ID addresses either a non-existing submodule or itself, the payload_ptr was larger than the packet (generates an early EOP), or the BCP header was larger than the header FIFO in the engine.
87-64	Tm_dlg_hw_timestamp	24 bits of a free running time stamp. The free running timer is in the TM.
63-32	Tm_dlg_sw_timestamp	The radio time stamp from a TM MMR register set by software
31-0	Global_hdr_field	32 bits from the global header will be used for software to identify the packet.

3.4.4 Example

An example of error reporting is shown in Figure 3-18.

Figure 3-18. Error Reporting Example



One packet into a submodule causes two errors (2 and 5). These two errors are both masked/disabled (shown as 'D') on DSP0, but at least one of them is enabled (shown as 'E') on each of the other DSPs. Therefore, DSP1, 2, and 3 have enabled error signals (shown as red arrows). The RX flow table entry selected in the packet's global header is setup to allow interrupts to DSPs 0 and 3. Based on this, an interrupt signal for DSP 3 is sent to the top level aggregation and then out of the BCP. It is assumed that no data logger event is present.

Another set of BCP registers (in the TM) that are used to enable a BCP global halt are setup to allow interrupts from DSPs 0 and 1 to halt the BCP. Since there are no interrupts on these two DSPs, the BCP is not halted. In this case the DSP could still put the data logger into a hold state when the interrupt occurs so that although packets continue, the data logger state is frozen and can be read. If the data logger is in hold state, it is a global hold so no packets are written.

To clear the error, the DSP must clear the error at the submodule(s) that caused the error and then clear the interrupt through the end of interrupt interface. If the data logger was halted or the BCP was halted, these halts also need to be cleared.

3.5 Remote Use

The BCP can be used remotely through rapidIO or through the antenna interface packet mode. It can also be accessed through the Ethernet interface, though in this case, packets are limited to about 9500 bytes. There are no specific features in the BCP to support remote use except for the capability to pass through protocol-specific information to the output descriptor. The pass-through user-defined information field can be used with the Ethernet interface for the Ethernet header.

3.5.1 Remote Use of TAC

For WCDMA downlink remote usage, the output of the BCP is sent to a remote device with the TAC (Transmit Accelerator). If sRIO type 9 is used, these type9 packet payloads will be deposited in some Navigator packet buffer. The next processing for this packet payload needs to be done by TAC. Due to the present organization of the TAC functional library, data for every WCDMA physical channel is expected to be organized in a linked list of uniform sized blocks of 336 bytes. More than one 336 byte block is necessary when the packet has more than 336 bytes. There exists a software memory handler which manages free block pools, allocation of blocks to channels, etc. Each block further has a header section where vital TAC specific information is embedded. This has, among other things, a link to the next block.

Since the above block-based memory organization does not match natively the data organization for Navigator buffers the software in the device containing the TAC will have to reorganize (perform data-movement on) the packet payload data to conform to the block based organization (along with appropriate headers) expected by TAC. The BCP data output format is compatible with the TAC, but the BCP does not add the TAC header and does not break the data into TAC blocks.

3.5.2 Remote Use of TCP3D

After rate de-matching and LLR combining, the BCP sends code blocks to a queue. This could be directed over type9 sRIO or could be a local queue. It will be up to the TCP3d driver outside of the BCP to send these code blocks to the TCP3D. The RD module places a header on each packet as described in the RD submodule section.

Submodules and Registers

This chapter describes the BCP submodules and their associated registers.

Topic **Page**

4.1 BCP Submodules

This section describes the memory mapped registers and packet header configuration fields for each BCP submodule.

4.1.1 Top-level Memory Map and MOD_IDs

Table 4-1 shows the top-level memory map of the BCP. Each submodule has a base offset that is relative to the base address of the BCP (given in the chip's device data sheet). Each submodule also has a data logger memory (DLG RAM) that is listed in the table as well.

Table 4-1. BCP Top-Level Memory Map and MOD_IDs

Module	Hex addr start	Hex addr end	MOD_ID
TM (includes BCP top-level regs)	0	3FF	0
DIO	400	4FF	13
INT	500	5FF	4
RM	600	6FF	5
ENC	700	7FF	6
MOD	800	8FF	7
DNT	900	9FF	12
SSL	A00	AFF	9
RD	B00	BFF	10
COR	C00	CFF	11
CRC	D00	DFF	8
TM DLG RAM	2000	2FFF	N/A
DIO DLG RAM	3000	3FFF	N/A
INT DLG RAM	4000	4FFF	N/A
RM DLG RAM	5000	5FFF	N/A
ENC DLG RAM	6000	6FFF	N/A
MOD DLG RAM	7000	7FFF	N/A
DNT DLG RAM	8000	8FFF	N/A
SSL DLG RAM	9000	9FFF	N/A
RD DLG RAM	A000	AFFF	N/A
COR DLG RAM	B000	BFFF	N/A
CRC DLG RAM	C000	CFFF	N/A
Packet DMA	10000	1FFFF	N/A

4.2 Traffic Manager (TM) (Includes BCP Top-level Registers)

All packets going into and out of the BCP pass through the Traffic Manager (TM). The TM decides from which queues to accept input data (priorities), reformats data (bit/byte/word swizzling) and handles endianness, manages the destination of output packets through the flow table, creates the output packets, and supports some special features like "flush," "drop," and "many-to-one" packet concatenation. It also contains the BCP top-level registers.

4.2.1 Memory Mapped Registers

The TM block has a set of data logger registers as described in the data logger section. The rest of the TM registers are described below. All fields are R/W unless otherwise stated. Reserved bits are always read-only and defaulted to 0.

Table 4-2. Traffic Manager Memory Map

Register Name	Address relative to TM base address	Description
BCP_PID	0x000	The Peripheral ID (PID) register contains the major and minor revision numbers for the module.
BCP_SOFT_RST	0x004	Allows the software to tear down the RX and TX paths as well as perform a full reset of the BCP.
TM_HALT_CTRL	0x008	This register is used to re-enable the BCP after a halt has occurred.
TM_HALT_STATUS	0x00C	This register is used to check if the BCP is halted. It also shows which TX queue received the packet that caused that halt.
BCP_EMU_CLKSTOP_CTRL	0x010	This register is used to control the behavior of the BCP when emulation suspend is active. The BCP can be setup to stop or to keep running during emulation suspend.
BCP_EMU_CLKSTOP_STATUS	0x014	This register is used to check if the submodules in the BCP are idle or not.
BCP_DLG_SW_TIMESTAMP	0x018	Allows the software to set a 32-bit value that is used by all the BCP submodules dataloggers when they record an entry in their data logger RAMs.
BCP_DLG_HW_TIMESTAMP	0x01C	Allows the software to read the current hardware timestamp counter value.
BCP_DLG_HW_TIMESTAMP_CTRL	0x020	This register is used to enable the free-running hardware timestamp counter inside the BCP. The current hardware timestamp is recorded by the data logger for each entry in the data logger RAM.
BCP_DLG_HOLD_RST	0x024	Allows the software to put the data loggers in DLG_IDLE or DLG_HOLD status regardless of the data logger control register settings for synchronizing starting and holding of data.
TM_CONTROL	0x028	This register can be used to set the src_id in the output packet descriptors, disable BCP data processing for Packet DMA loopback testing, and disable the automatic raising of priority for TX Packet DMA paths with active packets. Please see the queue mapping and prioritization section for more information about prioritization.
TM_TX_CDMAHP_RD_PRIORITY	0x02C	This register is used to control the TX Packet DMA Read Arbitration. Please see the queue mapping and prioritization section for more information about prioritization.
TM_TX_QFIFO_RD_DEST_SEL	0x030	This register is used to select priorities for the TX QFIFO read arbiter and also control the mapping of TX QFIFO to TX BSS port. Please see the queue mapping and prioritization section for more information about prioritization.
TM_RX_QUEUE[3:0]_FIFO_WR_ARB_PRIORITY	0x034	This register is used to control the RX QFIFO Write Arbitration for four of the RX QFIFOs. Please see the queue mapping and prioritization section for more information about prioritization.
TM_RX_QUEUE[7:4]_FIFO_WR_ARB_PRIORITY	0x038	This register is used to control the RX QFIFO Write Arbitration for four of the RX QFIFOs. Please see the queue mapping and prioritization section for more information about prioritization.
MMR_RX_CDMA_WR_ARB_PRIORITY	0x03C	This register is used to select priorities for the RX Packet DMA Write Arbitration. Please see the queue mapping and prioritization section for more information about prioritization.
BCP_INTERRUPT_0_STATUS	0x040	This is a read only interrupt status register indicating which module interrupted DSP 0. The interrupts must be cleared in the submodule interrupt registers.
BCP_0_HALT_ON_ERROR	0x044	This register controls whether the BCP halts when a specified submodule interrupts DSP 0. There is also a bit that allows the DSP to force a BCP halt.
BCP_INTERRUPT_1_STATUS	0x048	This is a read only interrupt status register indicating which module interrupted DSP 1. The interrupts must be cleared in the submodule interrupt registers.
BCP_1_HALT_ON_ERROR	0x04C	This register controls whether the BCP halts when a specified submodule interrupts DSP 1. There is also a bit that allows the DSP to force a BCP halt.

Table 4-2. Traffic Manager Memory Map (continued)

Register Name	Address relative to TM base address	Description
BCP_INTERRUPT_2_STATUS	0x050	This is a read only interrupt status register indicating which module interrupted DSP 2. The interrupts must be cleared in the submodule interrupt registers.
BCP_2_HALT_ON_ERROR	0x054	This register controls whether the BCP halts when a specified submodule interrupts DSP 2. If the BCP is halted due to an error interrupt, it will resume operation after the interrupt source is cleared in the submodule. There is also a bit in this register that allows the DSP to force a BCP halt.
BCP_INTERRUPT_3_STATUS	0x058	This is a read only interrupt status register indicating which module interrupted DSP 3. The interrupts must be cleared in the submodule interrupt registers.
BCP_3_HALT_ON_ERROR	0x05C	This register controls whether the BCP halts when a specified submodule interrupts DSP 3. There is also a bit that allows the DSP to force a BCP halt.
BCP_INTERRUPT_EOI	0x060	This register must be written by the DSP after finishing the interrupt processing to allow further interrupts to occur.
CDMA_DESC_STARVE_STATUS	0x064	Read only status bits of CDMAHP channels that had a descriptor starvation on RX.
CDMA_DESC_STARVE_CLR	0x068	Forces a clear of the descriptor status bit in the Cdmahp_desc_starve_status register.
CDMA_DESC_STARVE_SET	0x06C	Forces a set of the descriptor status bit in the Cdmahp_desc_starve_status register.
CDMA_DESC_STARVE_INTR_SEL	0x070	Selects which DSP gets interrupted for each channel. Each channel can select any of the 4 DSPs to interrupt if it is starved.
TM_FLOW_ID_TABLE[63:0]	0x100 -> 0x1FC	This table has a 32-bit register for each BCP RX flow ID. The registers contain control parameters that will be applied to packets that use the corresponding flow ID. These parameters include the protocol-specific flags and PKT_TYPE for the RX descriptor, the output QFIFO to be used, the DSP cores that may receive error interrupts, and the input and output data format and endian conversion settings.

Table 4-3. TM Register Details

MMR Register Name	Description		
BCP_PID	The revision of the BCP is kept in the TM and is read only.		
	Bits	**Name**	**Description**
	31-30	PID_SCHEME	Scheme that this register is compliant with
	29-28	Reserved	Reads return 0 and writes have no effect
	27-16	PID_FUNC	Function
	15-11	PID_RTL	RTL revision
	10-8	PID_MAJOR	Major revision
	7-6	PID_CUSTOM	Custom revision
	5-0	PID_MINOR	Minor revision

Table 4-3. TM Register Details (continued)

MMR Register Name	Description		
BCP_SOFT_RST	Control bits for doing a clean soft reset of the BCP.		
	Bits	**Name**	**Description**
	31-3	Reserved	Reserved
	2	TX_TEARDOWN	TX path stops passing PKTDMA data to the QFIFOs. Any PKTDMA TX queue that has data will be read with the data dropped so the PKTDMA can process the full packet.
	1	RX_TEARDOWN	RX path stops processing any rx packets when set. If a port had an active packet it will send an EOP word to the QFIFO to finish off the packet to the PKTDMA.
	0	SOFT_RST	Self-clearing reset bit that resets the entire BCP other than the PKTDMA and VBUSM interfaces.
TM_HALT_CTRL	**Bits**	**Name**	**Description**
	31-8	Reserved	Reserved
	7-0	CLEAR_HALT	Clears the halt in the selected queue.
TM_HALT_STATUS	**Bits**	**Name**	**Description**
	31-8	Reserved	
	7-0	HALT_STATUS	Status of which queues requested a halt.
BCP_EMU_CLKSTOP_CTRL	**Bits**	**Name**	**Description**
	31-3	Reserved	Reserved
	2	EMU_RT_SEL	1 = emu_dbgsusp_rt 0 = Emu_dbgusp
	1	Reserved (soft_stop)	Always doing soft stop so this bit is not used.
	0	EMU_freerun	Ignores emu reqs when set.
BCP_EMU_CLKSTOP_STATUS	Read only status of the modules idle indicators.		
	Bits	**Name**	**Description**
	31-12	Reserved	Reserved
	11-0	IDLE	Bit- Name 11 = AND of {10:0} 10 = DNT 9 = RD 8 = SSL 7 = COR 6 = INT 5 = MOD 4 = RM 3 = ENC 2 = CRC 1 = DIO 0 = TM
BCP_DLG_SW_TIMESTAMP	32 bit value distributed to all data_loggers		
BCP_DLG_HW_TIMESTAMP	**Bits**	**Name**	**Description**
	31-24	Reserved	Reserved
	23-0	HW_TIMESTAMP	Read only counter providing a timestamp for data logging.
BCP_DLG_HW_TIMESTAMP_CTRL	**Bits**	**Name**	**Description**
	31-1	Reserved	Reserved
	0	HW_TIMESTAMP_RUN	Resets the timestamp to 0 when low. Timestamp increments by 1 every clock when high.

Table 4-3. TM Register Details (continued)

MMR Register Name	Description		
BCP_DLG_HOLD_RST	Provides control for putting data loggers in DLG_IDLE or DLG_HOLD regardless of the data_logger_ctl settings for synchronizing starting and holding of data.		
	Bits	**Name**	**Description**
	31-27	Reserved	Reserved
	26-16	HOLD	BIT- SUBMODULE 10 = DNT 9 = RD 8 =SSL 7 = COR 6 = INT 5 = MOD 4 = RM 3 = ENC 2 = CRC 1 = DIO 0 = TM
	15-11	Reserved	Reserved
	10-0	RESET	BIT- SUBMODULE 10 = DNT 9 = RD 8 =SSL 7 = COR 6 = INT 5 = MOD 4 = RM 3 = ENC 2 = CRC 1 = DIO 0 = TM
TM_CONTROL	**Bits**	**Name**	**Description**
	31-16	Reserved	Reserved
	15-8	CDMAHP_SRC_ID	Src_id for PKTDMA info word
	7-3	Reserved	Reserved
	2	CDMAHP_DISABLE	Masks off any signals from the PKTDMA if high. Typically only used for PKTDMA loopback.
	1	FRC_PAYLOAD_ENDIAN	Forces the payload to follow the endian conversion settings, regardless of whether the device endian bit is set to big-endian or not.
	0	MMR_TX_CDMAHP_ RD_ARB_HPRIORITY	Enables raising priority of TX PKTDMA paths with active packets.

Table 4-3. TM Register Details (continued)

MMR Register Name	Description		
TM_TX_CDMAHP_RD_PRIORITY	Selects priorities for the TX Packet DMA Read Arbitration.		
	Bits	**Name**	**Description**
	31-16	Reserved	Reserved
	15-14	MMR_TX_CDMAHP_RD_ARB_PRIORITY[7]	Priority for reading PKTDMA for QFIFO 7.
	13-12	MMR_TX_CDMAHP_RD_ARB_PRIORITY[6]	Priority for reading PKTDMA for QFIFO 6.
	11-10	MMR_TX_CDMAHP_RD_ARB_PRIORITY[5]	Priority for reading PKTDMA for QFIFO 5.
	9-8	MMR_TX_CDMAHP_RD_ARB_PRIORITY[4]	Priority for reading PKTDMA for QFIFO 4.
	7-6	MMR_TX_CDMAHP_RD_ARB_PRIORITY[3]	Priority for reading PKTDMA for QFIFO 3.
	5-4	MMR_TX_CDMAHP_RD_ARB_PRIORITY[2]	Priority for reading PKTDMA for QFIFO 2.
	3-2	MMR_TX_CDMAHP_RD_ARB_PRIORITY[1]	Priority for reading PKTDMA for QFIFO 1.
	1-0	MMR_TX_CDMAHP_RD_ARB_PRIORITY[0]	Priority for reading PKTDMA for QFIFO 0.RX
TM_TX_QFIFO_RD_DEST_SEL	Selects priorities for the TX QFIFO read arbiter and also has the TX queue destination selects for which BSS port the TX queue sends its data to.		
	Bits	**Name**	**Description**
	31-30	MMR_TX_QUEUE_DEST_SEL [7]	Destination select for QFIFO 7.
	29-28	MMR_TX_QUEUE_DEST_SEL [6]	Destination select for QFIFO 6.
	27-26	MMR_TX_QUEUE_DEST_SEL [5]	Destination select for QFIFO 5.
	25-24	MMR_TX_QUEUE_DEST_SEL [4]	Destination select for QFIFO 4.
	23-22	MMR_TX_QUEUE_DEST_SEL [3]	Destination select for QFIFO 3.
	21-20	MMR_TX_QUEUE_DEST_SEL [2]	Destination select for QFIFO 2.
	19-18	MMR_TX_QUEUE_DEST_SEL [1]	Destination select for QFIFO 1.
	17-16	MMR_TX_QUEUE_DEST_SEL [0]	Destination select for QFIFO 0.
	15-14	MMR_TX_QFIFO_RD_ARB_PRIORITY[7]	Priority for reading QFIFO 7.
	13-12	MMR_TX_QFIFO_RD_ARB_PRIORITY[6]	Priority for reading QFIFO 6.
	11-10	MMR_TX_QFIFO_RD_ARB_PRIORITY[5]	Priority for reading QFIFO 5.
	9-8	MMR_TX_QFIFO_RD_ARB_PRIORITY[4]	Priority for reading QFIFO 4.
	7-6	MMR_TX_QFIFO_RD_ARB_PRIORITY[3]	Priority for reading QFIFO 3.
	5-4	MMR_TX_QFIFO_RD_ARB_PRIORITY[2]	Priority for reading QFIFO 2.
	3-2	MMR_TX_QFIFO_RD_ARB_PRIORITY[1]	Priority for reading QFIFO 1.
	1-0	MMR_TX_QFIFO_RD_ARB_PRIORITY[0]	Priority for reading QFIFO 0.

Table 4-3. TM Register Details (continued)

MMR Register Name	Description		
TM_RX_QUEUE[3:0]_FIFO_ WR_ARB_PRIORITY	Sets priority for data from each BSS port for 4 of the 8 queues.		
	Bits	**Name**	**Description**
	31-30	MMR_RX_ QFIFO3_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 3 from BSS port 3
	29-28	MMR_ RX_ QFIFO3_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 3 from BSS port 2.
	27-26	MMR_ RX_ QFIFO3_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 3 from BSS port 1.
	25-24	MMR_ RX_ QFIFO3_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 3 from BSS port 0.
	23-22	MMR_ RX_ QFIFO2_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 2 from BSS port 3.
	21-20	MMR_ RX_ QFIFO2_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 2 from BSS port 2.
	19-18	MMR_ RX_ QFIFO2_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 2 from BSS port 1.
	17-16	MMR_ RX_ QFIFO2_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 2 from BSS port 0.
	15-14	MMR_ RX_ QFIFO1_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 1 from BSS port 3.
	13-12	MMR_ RX_ QFIFO1_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 1 from BSS port 2.
	11-10	MMR_ RX_ QFIFO1_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 1 from BSS port 1.
	9-8	MMR_ RX_ QFIFO1_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 1 from BSS port 0.
	7-6	MMR_ RX_ QFIFO0_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 0 from BSS port 3.
	5-4	MMR_ RX_ QFIFO0_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 0 from BSS port 2.
	3-2	MMR_ RX_ QFIFO0_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 0 from BSS port 1.
	1-0	MMR_ RX_ QFIFO0_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 0 from BSS port 0.

Table 4-3. TM Register Details (continued)

MMR Register Name	Description		
TM_RX_QUEUE[7:4]_FIFO_WR_ARB_PRIORITY	Sets priority for data from each BSS port for 4 of the 8 queues.		
	Bits	Name	Description
	31-30	MMR_RX_ QFIFO7_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 7 from BSS port 3.
	29-28	MMR_ RX_ QFIFO7_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 7 from BSS port 2.
	27-26	MMR_ RX_ QFIFO7_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 7 from BSS port 1.
	25-24	MMR_ RX_ QFIFO7_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 7 from BSS port 0.
	23-22	MMR_ RX_ QFIFO6_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 6 from BSS port 3.
	21-20	MMR_ RX_ QFIFO6_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 6 from BSS port 2.
	19-18	MMR_ RX_ QFIFO6_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 6 from BSS port 1.
	17-16	MMR_ RX_ QFIFO6_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 6 from BSS port 0.
	15-14	MMR_ RX_ QFIFO5_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 5 from BSS port 3.
	13-12	MMR_ RX_ QFIFO5_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 5 from BSS port 2.
	11-10	MMR_ RX_ QFIFO5_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 5 from BSS port 1.
	9-8	MMR_ RX_ QFIFO5_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 5 from BSS port 0.
	7-6	MMR_ RX_ QFIFO4_WR_ ARB_PRIORITY [3]	Priority for writing QFIFO 4 from BSS port 3.
	5-4	MMR_ RX_ QFIFO4_WR_ ARB_PRIORITY [2]	Priority for writing QFIFO 4 from BSS port 2.
	3-2	MMR_ RX_ QFIFO4_WR_ ARB_PRIORITY [1]	Priority for writing QFIFO 4 from BSS port 1.
	1-0	MMR_ RX_ QFIFO4_WR_ ARB_PRIORITY [0]	Priority for writing QFIFO 4 from BSS port 0.
MMR_RX_CDMA_WR_ARB_ PRIORITY	Selects priorities for RX Packet DMA Write Arbitration.		
	Bits	**Name**	**Description**
	31-16	Reserved	Reserved
	15-14	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[7]	Priority for writing PKTDMA for queue 7.
	13-12	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[6]	Priority for writing PKTDMA for queue 6.
	11-10	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[5]	Priority for writing PKTDMA for queue 5.
	9-8	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[4]	Priority for writing PKTDMA for queue 4.
	7-6	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[3]	Priority for writing PKTDMA for queue 3.
	5-4	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[2]	Priority for writing PKTDMA for queue 2.
	3-2	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[1]	Priority for writing PKTDMA for queue 1.
	1-0	MMR_TX_CDMAHP_WR_ ARB_PRIORITY[0]	Priority for writing PKTDMA for queue 0.

Table 4-3. TM Register Details (continued)

MMR Register Name	Description		
BCP_INTERRUPT_ [3:0]_STATUS	Read only interrupt status indicating which module interrupted the processor. Bits can be set or cleared by the status_clr and status_set registers.		
	Bits	**Name**	**Description**
	31-11	Reserved	Reserved
	10-0	STATUS	BIT- SUBMODULE 10 = DNT 9 = RD 8 =SSL 7 = COR 6 = INT 5 = MOD 4 = RM 3 = ENC 2 = CRC 1 = DIO 0 = TM
BCP_[3:0]_HALT_ON_ERROR	**Bits**	**Name**	**Description**
	31-12	Reserved	Reserved
	11	FORCE_HALT	Forces a halt.
	10-0	HALT_ON_ERROR	Same as bcp_interrupt[3:0] status
BCP_INTERRUPT_EOI	**Bits**	**Name**	**Description**
	31-8	Reserved	Reserved
	7-0	BCP_INTERRUPT_EOI	Interrupt vector written by DSP at end of interrupt processing. Value to write is 0.
TM_FLOW_ID_TABLE[63:0]	Flow_id lookup table. 64 entries.		
	Bits	**Name**	**Description**
	31-28	Reserved	Reserved
	27-24	PS_FLAGS	Ps_flags field in packet info word 0.
	23-21	QFIFO_OUT	Output QFIFO
	20-18	FORMAT_OUT	Output data format (same as format_in)
	17-16	ENDIAN_OUT	Output data endian (same as endian_in)
	15-12	DSP_INT_SEL	Selects which of up to 4 DSPs to interrupt in case of error or interrupt event related to this packet. Each bit is one DSP. More than one bit can be set.
	11-7	PKT_TYPE	Pkt_type to put into info word 0 in RX.
	6-5	Reserved	Reserved
	4-2	FORMAT_IN	Input data format. 0 = No Change 1 = Reverse 32 bit words 2 = Reverse 16 bit words 3 = Reverse 8 bit words 4 = Reserved 5 = Swap 32 bit words 6 = Swap 16 bit words 7 = Swap 8 bit words
	1-0	ENDIAN_IN	Input data endian. 0 = 128 bit words 1 = 32 bit words 2 = 16 bit words 3 = 8 bit words

4.2.1.1 BCP CDMA Descriptor Starvation Interrupts (Available in Keystone-II Devices)

4.2.1.1.1 CDMA_DESC_STARVE_STATUS Register

Read only status bits of CDMAHP channels that had a descriptor starvation on RX.

Figure 4-1. CDMA_DESC_STARVE_STATUS Register

31	8	7	0
Reserved		Starve_Status	

Table 4-4. CDMA_DESC_STARVE_STATUS Register Details

Bits	Field	Description
31-8	Reserved	Reserved
7-0	Starve_Status	Starvation Status bits [7:0]

4.2.1.1.2 CDMA_DESC_STARVE_CLR Register

Forces a clear of the descriptor status bit in the Cdmahp_desc_starve_status register.

Figure 4-2. CDMA_DESC_STARVE_CLR Register

31	8	7	0
Reserved		Starve_clr	

Table 4-5. CDMA_DESC_STARVE_CLR Register Details

Bits	Field	Description
31-8	Reserved	Reserved
7-0	Starve_clr	Forces a clear of the descriptor status bit in the CDMAHP_DESC_STARVE_STATUS register.

4.2.1.1.3 CDMA_DESC_STARVE_SET Register

Forces a set of the descriptor status bit in the Cdmahp_desc_starve_status register.

Figure 4-3. CDMA_DESC_STARVE_SET Register

31	8	7	0
Reserved		Starve_set	

Table 4-6. CDMA_DESC_STARVE_SET Register Details

Bits	Field	Description
31-8	Reserved	Reserved
7-0	Starve_set	Forces a set of the descriptor status bit in the CDMAHP_DESC_STARVE_STATUS register.

4.2.1.1.4 CDMA_DESC_STARVE_INTR_SEL Register

Selects which DSP gets interrupted for each channel. Each channel can select any of the 4 DSPs to interrupt if it is starved.

Figure 4-4. CDMA_DESC_STARVE_INTR_SEL Register

31 28	27 24	23 20	19 16
Starve_intr_sel_7	Starve_intr_sel_6	Starve_intr_sel_5	Starve_intr_sel_4

15 12	11 8	7 4	3 0
Starve_intr_sel_3	Starve_intr_sel_2	Starve_intr_sel_1	Starve_intr_sel_0

Table 4-7. CDMA_DESC_STARVE_INTR_SEL Register Details

Bits	Field	Description
31-28	Starve_intr_sel_7	4-bit mask of which CPUs to interrupt.

Table 4-7. CDMA_DESC_STARVE_INTR_SEL Register Details (continued)

Bits	Field	Description
27-24	Starve_intr_sel_6	4-bit mask of which CPUs to interrupt.
23-20	Starve_intr_sel_5	4-bit mask of which CPUs to interrupt.
19-16	Starve_intr_sel_4	4-bit mask of which CPUs to interrupt.
15-12	Starve_intr_sel_3	4-bit mask of which CPUs to interrupt.
11-8	Starve_intr_sel_2	4-bit mask of which CPUs to interrupt.
7-4	Starve_intr_sel_1	4-bit mask of which CPUs to interrupt.
3-0	Starve_intr_sel_0	4-bit mask of which CPUs to interrupt.

4.2.2 Packet Header Configuration Fields

4.2.2.1 BCP Global Header

The global header (Table 4-8 and Table 4-9) is always the first 64 bits of the first 128-bit quad-word of any packet. It contains 32 bits of control information and 32 bits used to identify the packet in the data logger. The remaining 64 bits of the first word of any packet is the local header of the engine the packet is entering as well as the first word of configuration data for that engine.

Figure 4-5. Global Header Word 0

31 30	29 24	23 16	15 11	10 8
Reserved	FLOW_ID	GHDR_END_PTR	Reserved	RADIO_STANDARD

7 6	5	4	3	2	1 0
Reserved	HALT	DROP	FLUSH	Reserved	PKT_TYPE

Table 4-8. Global Header Word 0 Details

Bits	Field Name	Description
31-30	Reserved	Reserved
29-24	FLOW_ID	Used to index flow tables in traffic manager and Packet DMA. Can be used by any engine if needed.
23-16	GHDR_END_PTR	Number of 32-bit words in BCP header. Includes this word (Word 0) of the global header as well as all the local headers, the PS words, and user-defined info words in the TM local header. Data payload begins on the next 128-bit word boundary.
15-11	Reserved	Reserved
10-8	RADIO_ STANDARD	Radio Standard • 0 = LTE • 2 = WiMAX 902.16e • 4 = WCDMA/TD-SCDMA (Rel-99/HSPA)
7-6	Reserved	Reserved
5	HALT	When set to 1, generates a soft stop with an interrupt (if not masked) when the BCP goes idle. All currently active packets in the TX complete sending to their EOP. Halt is cleared by software writing to a register.
4	DROP	Drops packet. Used to in combination with FLUSH to fill PKTDMA FIFO with unused data so FLUSH does not end with stale data in PKTDMA FIFO.
3	FLUSH	Stops accepting new packets for the current TX QFIFO until the PKTDMA finishes writing the resulting output RX packet back into memory.
2	Reserved	Reserved

Table 4-8. Global Header Word 0 Details (continued)

Bits	Field Name	Description
1-0	PKT_TYPE	Indicates packet type: • 0 = NORM - Normal packet • 1 = SOM - Start of multi-packet packet • 2 = MOM - Middle of multi-packet packet • 3 = EOM – End of packet of a multi-packet packet

Figure 4-6. Global Header Word 1

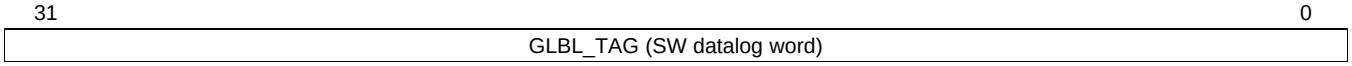


Table 4-9. Global Header Word 1 Details

Bits	Field Name	Description
31-0	GLBL_TAG (SW datalog word)	Global Tag (a.k.a. SW datalog word). Value written into each submodule's data logger memory when that submodule finishes outputting the processed data (if data logger active). NOTE: Lower 16 bits are used for dst_tag in RX packet descriptor.

4.2.2.2 TM Local Header

The TM local header specifies the length of the optional protocol-specific (PS) info and user-defined information fields that may be present in the packet.

Figure 4-7. TM Local Header Word 0

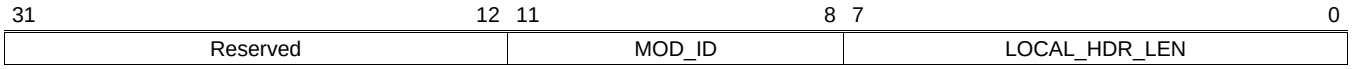


Table 4-10. TM Local Header Word 0 Details

Bits	Field Name	Description
31-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0). Note that this length does include the user-defined info and protocol-specific data which are part of the TM local header.

Figure 4-8. TM Local Header Word 1

Table 4-11. TM Local Header Word 1 Details

Bits	Field Name	Description
31-9	Reserved	Reserved
8-4	INFO_DATA_SIZE	Number of 128-bit quad-words of user-defined information. Max of 31.
3-0	PS_DATA_SIZE	Number of 32-bit words of protocol-specific (PS) data. Can be 0 to 8. Note that padding must be added to end of PS data if needed to make it a multiple of 4 words. This padding is not included in this size field.

The TM local header can optionally contain protocol-specific data and/or user-defined info starting at Word 2 (and going all the way up to Word 133). These fields are described in Section 3.1.2.

4.2.3 Error Bit Definitions

The TM error bits are defined in Table 4-12. These are the errors bits that will be recorded in the TM data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-12. TM Error Bit Definitions

TM Error bit definitions in Keystone-I		TM Error bit definitions in Keystone-II	
BIT	**Description**	**BIT**	**Description**
		7	Packet DMA (CDMA) starved
6	Bad_first_route. This error condition should only be caused by an illegal first route. This means that the MOD_ID for the first BCP submodule listed in the BCP header was an invalid MOD_ID. The only invalid MOD_IDs are 14 and 15.	6	Bad_first_route. This error condition should only be caused by an illegal first route. This means that the MOD_ID for the first BCP sub-module listed in the BCP header was an invalid MOD_ID. The only invalid MOD_IDs are 14 and 15.
5	Missing_eom or som or mom last byte count != 0x10 during their eop. This error is generated due to a missing end of “many-to-one” packet or the condition that the start or middle of a “many-to-one” packet was not a multiple of 16 bytes.	5	Missing_eom. Missing end of “many-to-one” packet.
4	Missing_som. Missing start of “many-to-one” packet.	4	Missing_som. Missing start of “many-to-one” packet.
3	Missing_eop. Missing end-of-packet.	3	som or mom xcnt != 0x10 during their eop. This error is generated when the condition at the start or middle of a “many-to-one” packet is not a multiple of 16 bytes.
2	Missing_sop. Missing start-of-packet.	2	Reserved

Table 4-12. TM Error Bit Definitions (continued)

TM Error bit definitions in Keystone-I		TM Error bit definitions in Keystone-II	
BIT	**Description**	**BIT**	**Description**
1	Dlg_done_int. Data logger done event.	1	Dlg_done_int. Data logger done event.
0	Halt_interrupt. Indicates that BCP is halted due to a halt packet or error interrupt.	0	Halt_interrupt. Indicates that BCP is halted due to a halt packet or error interrupt.

4.3 CRC

This submodule is used to perform the CRC and scrambling functions of WCDMA and TD-SCDMA and a transport block CRC for LTE and burst CRC for WiMAX. For HS-DSCH encoding and E-DCH recoding it is used once, while for Rel-99 voice and data channels it is used once in the transport channel based processing for CRC attachment and once again in the physical channel based processing . The physical channel based processing performs transport channel concatenation (removal of padding between transport channels) for Rel-99 FDD. It performs bit scrambling for Rel-99 TDD.

The input packet to the CRC block is one TTI of concatenated transport blocks. Each transport block will have its own CRC appended with the same polynomial. The CRC calculation is reset between transport blocks. In WCDMA mode, CRC bits are appended in bit-reversed order whereas in LTE mode bits are appended in generated order.

The CRC values are xor-ed with a user-configurable mask referred to as “method2.” The userId for method2 is sent in the method2 field. If method2 CRC generation is not used this value should be 0. This mask is used for method 2 for HS-DSCH [25.212, 4.5.1.2] (24-bit CRC) and for LTE PDCCH [36.212, 5.3.3.2] (16-bit CRC).

If no transport blocks are input to the CRC calculation (blocks = 0), no CRC attachment shall be done. If transport blocks are input to the CRC calculation (blocks > 0) and the size of a transport block is 0 (blklength = 0), CRC shall be attached, i.e., all parity bits equal to 0.

The scrambler works on the stream of concatenated blocks or channels and its sequence is reset at the beginning of each packet.

The input blocks may nominally be either 1-bit Data or 2-bit Data+DTX per sample. It is not required to CRC or scramble 2-bit data. With DTX inputs, a CRC is not calculated. Filler bits are concatenated at the start.

CRC generation and scrambling are enabled or disabled as shown in the table below. Up to six transport channels can be processed for Rel-99 physical processing. The transport channel header param sets are named for the Rel-99 transport channels (va, vb, vc, d1, d2, dc). Although they have specific names, there is no functional difference in the transport channel param sets. For other modes, there will only be one transport channel present and any of the param sets can be used (‘va’ is chosen in the table below).

Table 4-13. CRC Modes

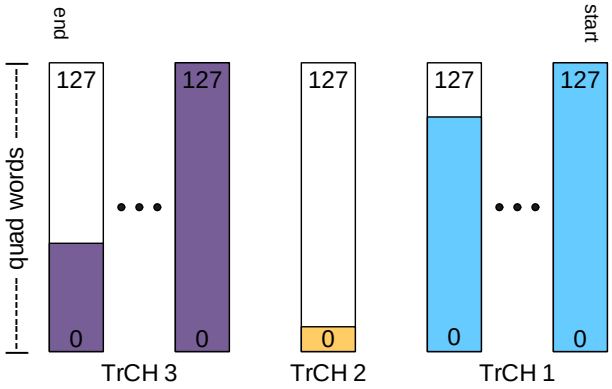
Standard	Input	Format	CRC	Scramble	Output	Transport Channel Header Params Used
LTE	Memory	1-bit	24-bit	No	ENC	va (can use any param set)
HS-DSCH	Memory	1-bit	24-bit	Yes	ENC	va (can use any param set)
E-DCH	Memory	1-bit	24-bit	No	ENC	va (can use any param set)
Rel-99 WCDMA/ TD-SCDMA Transport	Memory	1-bit	0/8/12/ 16/24	No	ENC	va (can use any param set)
Rel-99 WCDMA Physical	DIO	2-bit	0	No	INT	va, vb, vc, d1, d2, dc
Rel-99 TD-SCDMA Physical	RM	1-bit	0	Yes	INT	va, vb, vc, d1, d2, dc

4.3.1 Data Format

4.3.1.1 Input Data

Input Data is packed into 128-bit words. Each data value may have 1 or 2 input bits, in transport block concatenation it is always 1 bit. It is always 1 bit if CRCs are added or scrambling is enabled. Where 2-bit data is used the DTX bit will be the MS bit and the data bit the LS bit. The first data value to be processed is in the LSB of the first 128-bit word, and continues in packed format. Transport blocks are packed into a single continuous stream without data padding between blocks. Transport channels are padded and each transport channel starts on its own 128-bit alignment. When there is only one transport channel, it may contain more than one transport block. When more than one transport channel exists in the packet (Rel-99 downlink physical processing), each channel usually only contains a single data block. An example of the input format for the CRC submodule is shown in Figure 4-9.

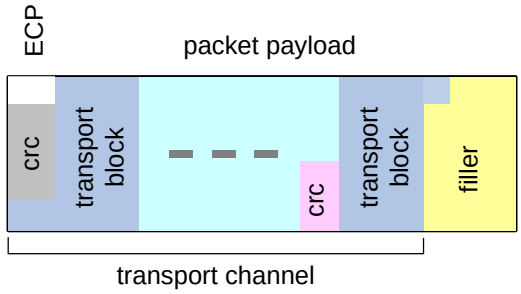
Figure 4-9. CRC Input Format



4.3.1.2 Output Data

Output data is always packed into 128-bit words, the first value to be transmitted is in the least significant bit(s). Transport blocks and transport channels are packed into a single continuous stream without data padding between blocks. Filler bits are inserted at the beginning of the packet payload. A n example of the output format for the CRC submodule with only one transport channel present in the packet is shown in Figure 4-10.

Figure 4-10. CRC Output Format



4.3.2 Memory Mapped Registers

The CRC module has a set of data logger registers as described in the data logger section. The rest of the CRC registers are described in the tables below. Please note that all the CRC registers are read-only and are only intended for debug.

4.3.2.1 CRC8_POLY

Figure 4-11. MMR: CRC8_POLY Register (offset 0x00)

31	24	23	0
crc8_poly		Reserved	
R - 0x9b		n/a	

Legend: R = Read only

Table 4-14. MMR: CRC8_POLY Register Details

Bit	Name	Description
31-24	crc8_poly	Value = any Polynomial for crc8
23-0	Reserved	Value = NA

4.3.2.2 CRC12_POLY

Figure 4-12. MMR: CRC12_POLY Register (offset 0x04)

31	20	19	0
crc12_poly		Reserved	
R - 0x80f		n/a	

Legend: R = Read only

Table 4-15. MMR: CRC12_POLY Register Details

Bit	Name	Description
31-20	crc12_poly	Value = any Polynomial for crc12
19-0	Reserved	Value = NA

4.3.2.3 CRC16_POLY

Table 4-16. MMR: CRC16_POLY Register Details

Bit	Name	Description
31-16	crc16_poly	Value = any Polynomial for crc16
15-0	Reserved	Value = NA

Figure 4-13. MMR: CRC16_POLY Register (offset 0x08)

31	16	15	0
crc16_poly		Reserved	
R - 0x1021		n/a	

Legend: R = Read only

4.3.2.4 CRC16W_POLY

Figure 4-14. MMR: CRC16W_POLY Register (offset 0x0C)

31	16	15	0
crc16w_poly		Reserved	
R - 0x8005		n/a	

Legend: R = Read only

Table 4-17. MMR: CRC16W_POLY Register Details

Bit	Name	Description
31-16	crc16w_poly	Value = any Polynomial for crc16w
15-0	Reserved	Value = NA

4.3.2.5 CRC24A_POLY

Figure 4-15. MMR: CRC24A_POLY Register (offset 0x10)

31	8	7	0
crc24a_poly		Reserved	
R - 0x864cfb		n/a	

Legend: R = Read only

Table 4-18. MMR: CRC24A_POLY Register Details

Bit	Name	Description
31-8	crc24a_poly	Value = any Polynomial for crc24a
7-0	Reserved	Value = NA

4.3.2.6 CRC24B_POLY

Figure 4-16. MMR: CRC24B_POLY Register (offset 0x14)

31	8	7	0
crc24b_poly		Reserved	
R - 0x800063		n/a	

Legend: R = Read only

Table 4-19. MMR: CRC24B_POLY Register Details

Bit	Name	Description
31-8	crc24b_poly	Value = any Polynomial for crc24b
7-0	Reserved	Value = NA

4.3.2.7 CRC32_POLY

Figure 4-17. MMR: CRC32_POLY Register (offset 0x18)

31	0
crc32_poly	
R - 0x0	

Legend: R = Read only

Table 4-20. MMR: CRC32_POLY Register Details

Bit	Name	Description
31-0	crc32_poly	Value = any Polynomial for crc32

4.3.2.8 CRC_INIT_SEL

Figure 4-18. MMR: CRC_INIT_VAL Register (offset 0x1C)

31	7	6	5	4	3	2	1	0
Reserved		crc32_init_val	crc24b_init_val	crc24a_init_val	crc16w_init_val	crc16_init_val	crc12_init_val	crc8_init_val
n/a		R - 0	R - 0	R - 0	R - 0	R - 0	R - 0	R - 0

Legend: R = Read only

Table 4-21. MMR: CRC_INIT_VAL Register Details

Bit	Name	Description
31-7	Reserved	Reserved
6	crc32_init_val	Initial value for crc32: • 1 = All 1 • 0 = All 0
5	crc24b_init_val	Initial value for crc24b: • 1 = All 1 • 0 = All 0
4	crc24a_init_val	Initial value for crc24a: • 1 = All 1 • 0 = All 0
3	crc16w_init_val	Initial value for crc16w: • 1 = All 1 • 0 = All 0
2	crc16_init_val	Initial value for crc16: • 1 = All 1 • 0 = All 0
1	crc12_init_val	Initial value for crc12: • 1 = All 1 • 0 = All 0
0	crc8_init_val	Initial value for crc8: • 1 = All 1 • 0 = All 0

4.3.3 Packet Header Configuration Fields

Figure 4-19 shows the mapping of the packet header configuration fields. Table 4-22 describes each field.

Figure 4-19. CRC Local Packet Header Word 0

31 26	25	24	23 16	15 12	11 8	7	6 0
Reserved	DTX_FORMAT	LTE_ORDER	FILLER_BITS	Reserved	MOD_ID	Reserved	LOCAL_HDR_LEN

Table 4-22. CRC Local Packet Header Word 0 Details

Bit	Name	Description
31-26	Reserved	Reserved
25	DTX_FORMAT	DTX Data Format: 1 = 2-bit data (DTX:data), 0 = 1-bit data (data only)
24	LTE_ORDER	Append CRC bits in LTE order: 1 = LTE order, 0 = WCDMA (reverse) order
23-16	FILLER_BITS	Number of filler bits (0s) to insert at the beginning of each packet payload. (0x00 – 0xFF)
15-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

Figure 4-20. CRC Local Packet Header Word 1

31 0
NUM_SCRAMBLE_SYS

Table 4-23. CRC Local Packet Header Word 1 Details

Bit	Name	Description
31-0	NUM_SCRAMBLE_SYS	Scrambler initialization value. Set to 0 to disable scrambling. For FDD Rel-99, CRC module does not perform scrambling for either stage, set NUM_SCRAMBLE_SYS to 0. For FDD HSDPA, CRC module performs CRC and bit scrambling. NUM_SCRAMBLE_SYS is the initial value of the scrambler, set it to 0x01000000 for the scrambling specified in 3GPP 25.212 section 4.5.1a. For TDD Rel-99, CRC module performs: • Only CRC for transport channel processing, set NUM_SCRAMBLE_SYS to 0. • Only bit scrambling for CCTrCH processing. NUM_SCRAMBLE_SYS is the initial value of the scrambler, set it to 0x01000000 for scrambling specified in 3GPP 25.222 section 4.2.9. For TDD HSDPA, CRC module performs CRC only, set NUM_SCRAMBLE_SYS to 0. Set to 0 for other standards.

Figure 4-21. CRC Local Packet Header Word 2

31 0
METHOD2_ID

Table 4-24. CRC Local Packet Header Word 2 Details

Bit	Name	Description
31-0	METHOD2_ID	CRC method 2 ID. If the CRC format is not crc0, the CRC result is XORed with this value. For CRC formats shorter than 32 bits, the value should be placed in the MSBs (most-significant-bits) of this register. METHOD2_ID comes from the 3GPP standard (3GPP 25.212 section 4.5.1.2 for FDD HSDPA) (3GPP 36.212 section 5.3.3.2 for LTE PDCCH). Set to 0 for other standards.

Figure 4-22. CRC Local Packet Header Word 3

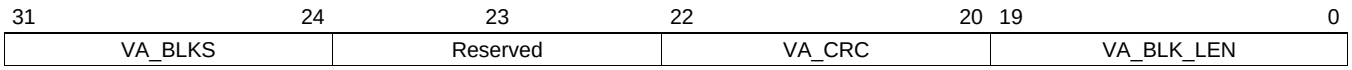


Table 4-25. CRC Local Packet Header Word 3 Details

Bit	Name	Description
31-24	VA_BLKS	Voice A Blocks: This value represents the number of transport blocks in this transport channel.
23	Reserved	Reserved
22-20	VA_CRC	Voice A CRC Format: 0 = crc0, 1 = crc8, 2 = crc12, 3 = crc16, 4 = crc16w, 5 = crc24a, 6 = crc24b, 7 = crc32
19-0	VA_BLK_LEN	Voice A Transport Block Length: This value represents the number of data bits in each transport block of this transport channel. When DTX_FORMAT = 1, represents number of 2-bit values.

Figure 4-23. CRC Local Packet Header Word 4

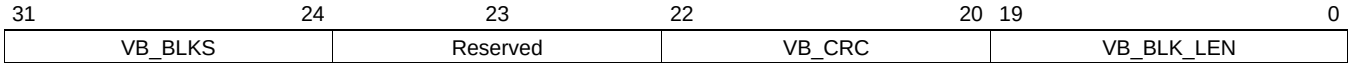


Table 4-26. CRC Local Packet Header Word 4 Details

Bit	Name	Description
31-24	VB_BLKS	Voice B Blocks: This value represents the number of transport blocks in this transport channel.
23	Reserved	Reserved
22-20	VB_CRC	Voice B CRC Format: 0 = crc0, 1 = crc8, 2 = crc12, 3 = crc16, 4 = crc16w, 5 = crc24a, 6 = crc24b, 7 = crc32
19-0	VB_BLK_LEN	Voice B Transport Block Length: This value represents the number of data bits in each transport block of this transport channel. When DTX_FORMAT = 1, represents number of 2-bit values.

Figure 4-24. CRC Local Packet Header Word 5

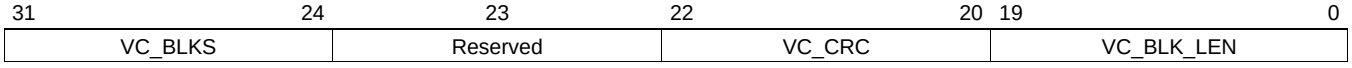


Table 4-27. CRC Local Packet Header Word 5 Details

Bit	Name	Description
31-24	VC_BLKS	Voice C Blocks: This value represents the number of transport blocks in this transport channel.
23	Reserved	Reserved
22-20	VC_CRC	Voice C CRC Format: 0 = crc0, 1 = crc8, 2 = crc12, 3 = crc16, 4 = crc16w, 5 = crc24a, 6 = crc24b, 7 = crc32
19-0	VC_BLK_LEN	Voice C Transport Block Length: This value represents the number of data bits in each transport block of this transport channel. When DTX_FORMAT = 1, represents number of 2-bit values.

Figure 4-25. CRC Local Packet Header Word 6

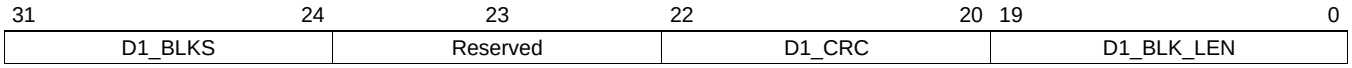


Table 4-28. CRC Local Packet Header Word 6 Details

Bit	Name	Description
31-24	D1_BLKS	Data 1 Blocks: This value represents the number of transport blocks in this transport channel.
23	Reserved	Reserved
22-20	D1_CRC	Data 1 CRC Format: 0 = crc0, 1 = crc8, 2 = crc12, 3 = crc16, 4 = crc16w, 5 = crc24a, 6 = crc24b, 7 = crc32
19-0	D1_BLK_LEN	Data 1 Transport Block Length: This value represents the number of data bits in each transport block of this transport channel. When DTX_FORMAT = 1, represents number of 2-bit values.

Figure 4-26. CRC Local Packet Header Word 7

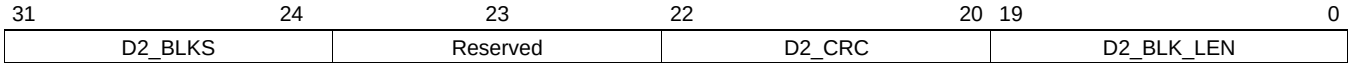


Table 4-29. CRC Local Packet Header Word 7 Details

Bit	Name	Description
31-24	D2_BLKS	Data 2 Blocks: This value represents the number of transport blocks in this transport channel.
23	Reserved	Reserved
22-20	D2_CRC	Data 2 CRC Format: 0 = crc0, 1 = crc8, 2 = crc12, 3 = crc16, 4 = crc16w, 5 = crc24a, 6 = crc24b, 7 = crc32
19-0	D2_BLK_LEN	Data 2 Transport Block Length: This value represents the number of data bits in each transport block of this transport channel. When DTX_FORMAT = 1, represents number of 2-bit values.

Figure 4-27. CRC Local Packet Header Word 8

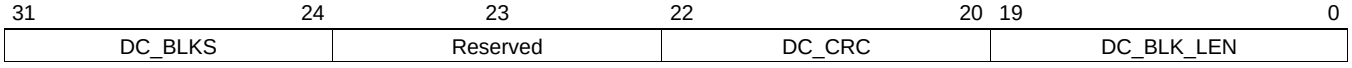


Table 4-30. CRC Local Packet Header Word 8 Details

Bit	Name	Description
31-24	DC_BLKS	DCCH Blocks: This value represents the number of transport blocks in this transport channel.
23	Reserved	Reserved
22-20	DC_CRC	DCCH CRC Format: 0 = crc0, 1 = crc8, 2 = crc12, 3 = crc16, 4 = crc16w, 5 = crc24a, 6 = crc24b, 7 = crc32
19-0	DC_BLK_LEN	DCCH Transport Block Length: This value represents the number of data bits in each transport block of this transport channel. When DTX_FORMAT = 1, represents number of 2-bit values.

4.3.4 Error Bit Definitions

The CRC error bits are defined in Table 4-31. These are the errors bits that will be recorded in the CRC data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-31. CRC Error Bit Definitions

BIT	NAME
6	Invalid packet header configuration. Detected: • multiple transport blocks with multiple TrCHs OR • packet with no TrCHs (all *_BLKS packet header fields equal to zero)
5	Invalid packet header configuration. Detected blk_len=0 with multiple TrCHs.
4	Invalid packet header configuration. Detected filler bits used with multiple TrCHs.
3	Invalid packet header configuration. Detected crc calculation used with multiple TrCHs.
2	Invalid packet header configuration. Detected crc12 used with multiple transport blocks.
1	Reserved
0	Reserved

4.4 ENC

The ENC submodule performs both Turbo encoding and convolutional encoding. It also performs code block scrambling for WiMAX 802.16e and code block CRC attachment for LTE. In the typical case, the input to the ENC block comes from the CRC submodule. The output goes to either rate matching or, for TD-SCDMA, to the interleaver.

The CRC polynomials and initialization values are programmable through memory mapped registers during initialization, but are preset to useful default values at reset.

Tail-biting for convolutional codes is dependent on the radio standard. For LTE, tail-biting is enabled. For WCDMA, there is no tail biting and there are eight tail bits per stream. Convolutional encoding is not supported in WiMAX.

4.4.1 Data Format

4.4.1.1 Input Data

The input to the ENC submodule is a packet containing one or more code blocks. The output is also a packet containing the same number of code blocks. This is true for convolutional or Turbo codes. The order of packets and of code blocks within a packet is preserved from input to output. Note that if the ENC submodule detects an invalid set of input parameters in its local configuration header, it will output a single code block with a code block size of 48 bits, report the error, and move on to the next input packet.

The input to the ENC is always in a one bit per bit format little endian. The BSS is a 128-bit interface. Bit 0 of the 128-bit BSS is considered the first bit in time and bit 127 is the last bit in time. If the input to the BCP is not in this format, the traffic manager will have converted it to this format. The ENC submodule does not do any format conversion.

Since code block s are packed with no padding between them, their boundaries are not necessarily on a 128-bit boundary – they can be on any bit boundary. The ENC submodule is responsible for segmenting the packet into code blocks.

The input to ENC is a packet with an integer number of code blocks. There can be one or more code blocks in a packet. There will be a field in the header that tells the ENC submodule the number of code blocks in the packet and there will be a list of lengths of each code block.

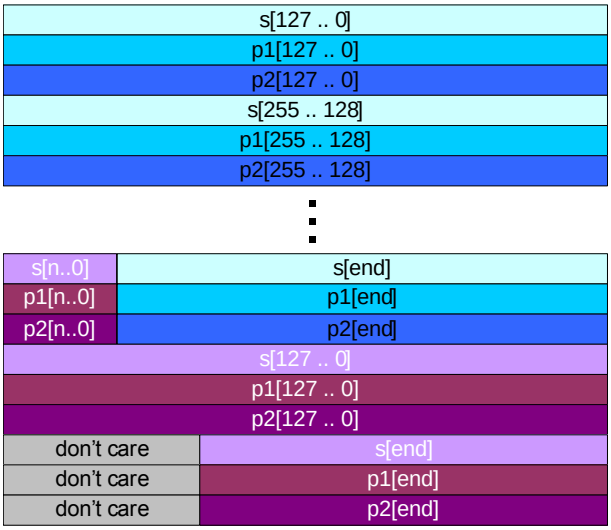
4.4.1.2 Output Data

The output format varies depending on the radio standard. As in the input, the least significant bit on the BSS, bit 0, will be the first output bit and bit 127 will be the last output bit.

There are two possible output formats for both the Turbo and convolutional encoder for WCDMA and LTE/WiMAX. The mode select bit in the configuration header will determine which output format is used.

In Figure 4-28 for WCDMA mode, the output alternates among systematic, parity1, and parity2 bits 128-bits at a time. The first code block in the packet always starts at bit 0 on the bus. Subsequent code blocks within a packet are packed onto the 128-bit bus. If the first code block ends in the middle of the 128-bit bus, the next code block will fill the remaining bits so that all 128-bits are used for every code block except possibly the last code block in the packet. The unused bits at the end are undefined and can be any value.

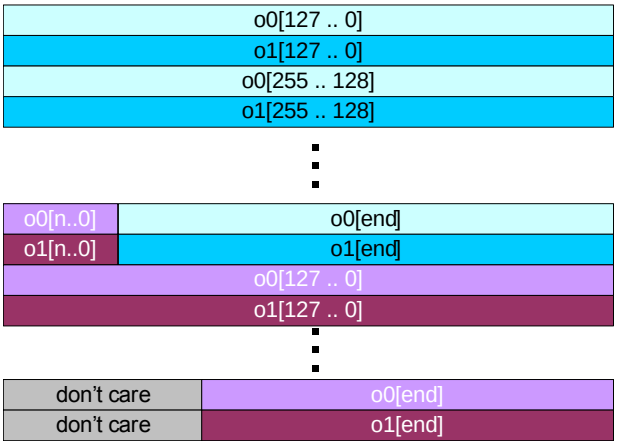
Figure 4-28. Output Format of the Rate 1/3 Convolutional Code and Turbo Code for WCDMA



NOTE: Don't care bits are output as valid data.

If the WCDMA rate 1/3 convolutional code is used, the output format is also as shown in the previous figure with o0, o1, and o2 (output0, output1, output2) replacing systematic, parity1, and parity2 respectively. If the WCDMA rate 1/2 convolutional code is used, the output format is as shown in the next figure. This is exactly like the rate 1/3 output format except that the o2 stream is missing. The unused bits at the end are undefined and can be any value.

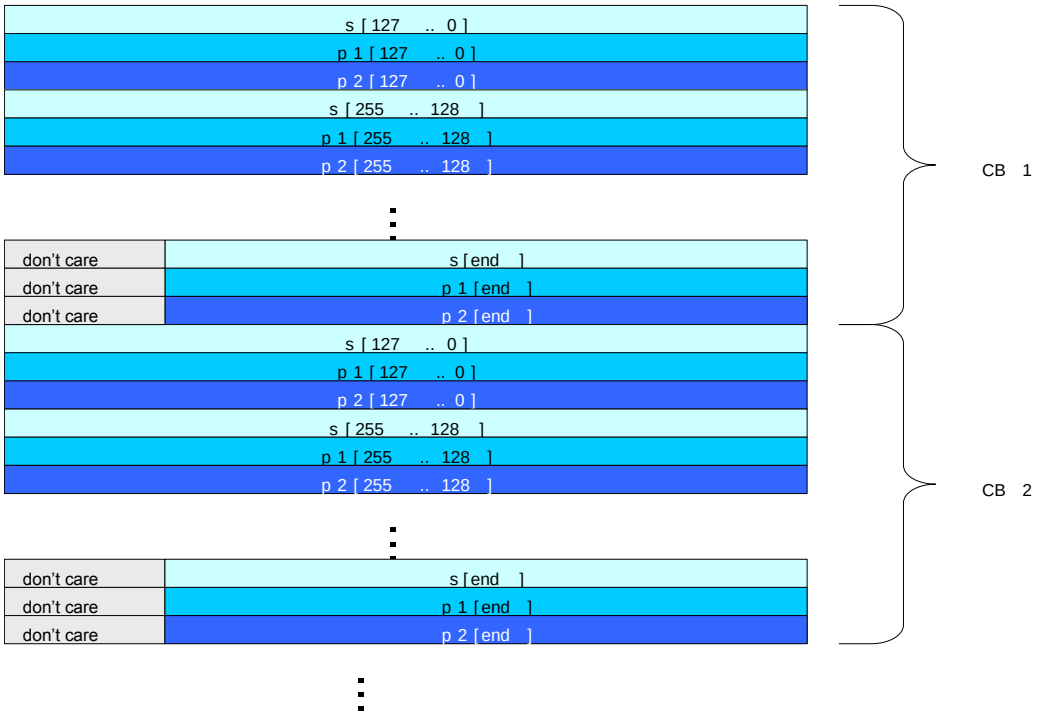
Figure 4-29. Output Format of the Rate 1/2 Convolutional Code



NOTE: Don't care bits are output as valid data.

In Figure 4-30 for LTE/WiMAX modes, the output is in the same format as for WCDMA except that from code block to code block, the output is not packed on the 128-bit bus. In this case, if the code block is not a multiple of 128-bits, the pad bits at the end of the codeblock are not used by rate matching and so the values do not matter. The unused bits at the end are undefined and can be any value.

Figure 4-30. Output Format of the Rate 1/3 Convolutional Code and Turbo Code for LTE/WiMAX



NOTE: The systematic, parity1 and parity 2 start on 128-bit boundaries. If the number of systematic, parity1, or parity2 bits is not a multiple of 128, the last 128-bit transmission for each set has unused don't care bits. Don't care bits are output as valid data.

In the case of the convolutional code, the systematic, parity1, and parity2 streams are replaced by d(0), d(1) and d(2) in the case of a 1/3 code – there is only d(0) and d(1) in the case of a 1/2 code.

In WiMAX mode, the output format of the encoder is similar to that shown in the last figure for LTE. However, in place of the systematic bits are the A and B bits interleaved (A is bit 0). The parity1 bits are replaced by Y1W1 (Y1 bit is bit 0) and the parity2 bits are replaced by Y2W2 (Y2 bit is bit 0). As in the figure, if the number of A and B bits is not a multiple of 128, there will be unused bits on the bus after the AB bits are output. The same is true for Y1W1 and Y2W2 bits.

4.4.2 Memory Mapped Registers

The ENC module has a set of data logger registers as described in the data logger section. The rest of the ENC registers are described in the tables below. The highlighted registers use the upper (most-significant) bits for the data fields. The other registers use the lower (least-significant) bits.

Table 4-32. ENC Memory Map

Name	Number of bits	Address offset	Access (R/W)	Reset value	Description
PolyCoef1	32	0x700	R/W	0x4D9413	Used for WCDMA Code Rate 1/2
PolyCoef2	32	0x704	R/W	0xD5AA5F	Used for WCDMA Code Rate 1/3
PolyCoef3	32	0x708	R/W	0xDDC3C0	Used for LTE Code Rate 1/3
CRC24Init0	24	0x714	R/W	0x00000000	CRC-24 polynomial (LTE)
CRC24Poly0	24	0x718	R/W	0x80006300	CRC-24 polynomial (LTE)
CRC16Init1	16	0x724	R/W	0x00000000	CRC-16 polynomial
CRC16Poly1	16	0x728	R/W	0x10210000	CRC-16 polynomial

4.4.3 Packet Header Configuration Fields

Figure 4-31. ENC Local Header Word 0

31 20	19	18 17	16	15 12	11 8	7	6 0
Reserved	CODE_RATE_FLAG	SCR_CRC_ENABLE	TURBO_CONV_SEL	Reserved	MOD_ID	Reserved	LOCAL_HDR_LEN

Table 4-33. ENC Local Header Word 0 Details

Bits	Field Name	Description
31-20	Reserved	Reserved
19	CODE_RATE_FLAG	For convolutional codes only (ignored if Turbo coding is enabled). • 0 = rate 1/3 • 1 = rate 1/2
18-17	SCR_CRC_ENABLE	Determines whether to scramble (SCR) and/or add code block CRC on all code blocks. • 0 = No SCR, No CRC • 1 = CRC24 (No SCR) (LTE) • 2 = SCR, CRC16 • 3 = SCR, No CRC (WiMAX 802.16e)
16	TURBO_CONV_SEL	Determines whether packet contains data to be Turbo or convolutionally encoded. One packet will never combine the two. • 0 = convolutional • 1 = Turbo
15-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

Figure 4-32. ENC Local Header Word 1

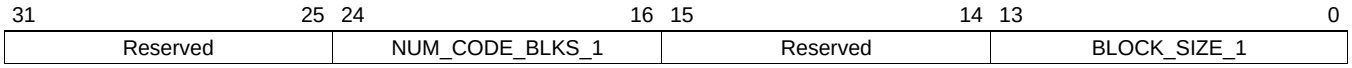


Table 4-34. ENC Local Header Word 1 Details

Bits	Field Name	Description
31-25	Reserved	Reserved
24-16	NUM_CODE_BLKS_1	number of consecutive code blocks of block size 1
15-14	Reserved	Reserved
13-0	BLOCK_SIZE_1	the size of block size 1 in bits (this size represents the total code block size including the code block CRC bits that are added by the ENC module when SCR_CRC_ENABLE = 1 or 2)

Figure 4-33. ENC Local Header Word 2

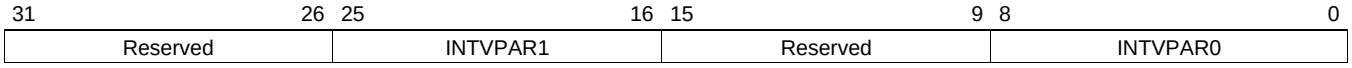


Table 4-35. ENC Local Header Word 2 Details

Bits	Field Name	Description
31-26	Reserved	Reserved
25-16	INTVPAR1	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: Interleaver parameters f2 in formula y(i) = (f1*i + f2*i^2) mod K • WiMAX mode: interleaver parameters P1
15-9	Reserved	Reserved
8-0	INTVPAR0	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: Interleaver parameters f1 in formula y(i) = (f1*i + f2*i^2) mod K • WiMAX mode: interleaver parameters P0

Figure 4-34. ENC Local Header Word 3

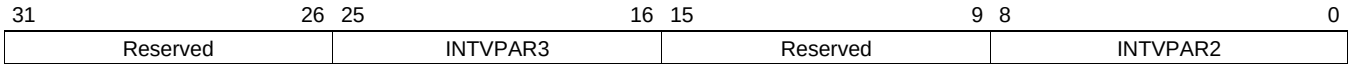


Table 4-36. ENC Local Header Word 3 Details

Bits	Field Name	Description
31-26	Reserved	Reserved
25-16	INTVPAR3	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: must be set to 0 • WiMAX mode: interleaver parameters P3
15-9	Reserved	Reserved
8-0	INTVPAR2	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: must be set to 0 • WiMAX mode: interleaver parameters P2

Figure 4-35. ENC Local Header Word 4

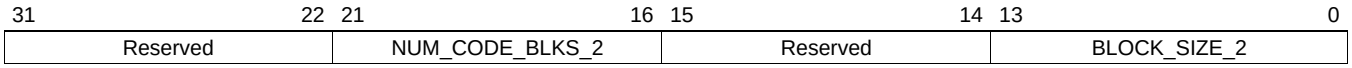


Table 4-37. ENC Local Header Word 4 Details

Bits	Field Name	Description
31-22	Reserved	Reserved
21-16	NUM_CODE_BLKS_2	number of consecutive code blocks of block size 2 (set to 0 if there is only one code block size)
15-14	Reserved	Reserved
13-0	BLOCK_SIZE_2	the size of block size 2 in bits (this size represents the total code block size including the code block CRC bits that are added by the ENC module when SCR_CRC_ENABLE = 1 or 2)

Figure 4-36. ENC Local Header Word 5

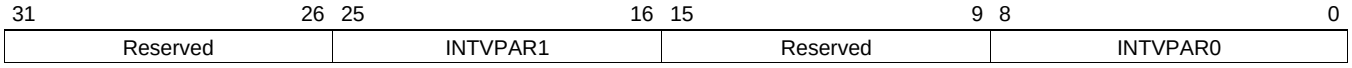


Table 4-38. ENC Local Header Word 5 Details

Bits	Field Name	Description
31-26	Reserved	Reserved
25-16	INTVPAR1	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: Interleaver parameters f2 in formula y(i) = (f1*i + f2*i^2) mod K • WiMAX mode: interleaver parameters P1
15-9	Reserved	Reserved
8-0	INTVPAR0	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: Interleaver parameters f1 in formula y(i) = (f1*i + f2*i^2) mod K • WiMAX mode: interleaver parameters P0

Figure 4-37. ENC Local Header Word 6

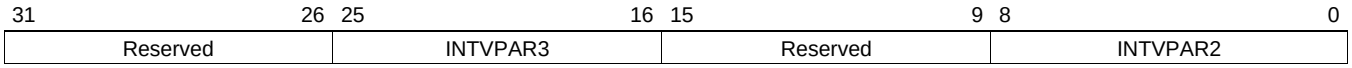


Table 4-39. ENC Local Header Word 6 Details

Bits	Field Name	Description
31-26	Reserved	Reserved
25-16	INTVPAR3	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: must be set to 0 • WiMAX mode: interleaver parameters P3
15-9	Reserved	Reserved
8-0	INTVPAR2	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: must be set to 0 • WiMAX mode: interleaver parameters P2

Figure 4-38. ENC Local Header Word 7

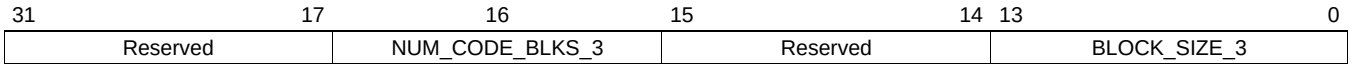


Table 4-40. ENC Local Header Word 7 Details

Bits	Field Name	Description
31-17	Reserved	Reserved
16	NUM_CODE_BLKS_3	number of consecutive code blocks of block size 3 (set to 0 if there are only one or two code block sizes)
15-14	Reserved	Reserved
13-0	BLOCK_SIZE_3	the size of block size 3 in bits (this size represents the total code block size including the code block CRC bits that are added by the ENC module when SCR_CRC_ENABLE = 1 or 2)

Figure 4-39. ENC Local Header Word 8

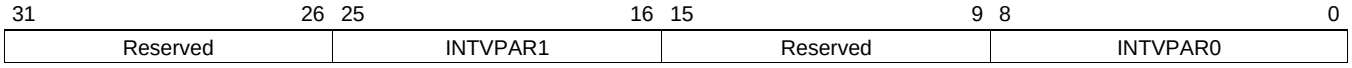


Table 4-41. ENC Local Header Word 8 Details

Bits	Field Name	Description
31-26	Reserved	Reserved
25-16	INTVPAR1	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: Interleaver parameters f2 in formula y(i) = (f1*i + f2*i^2) mod K • WiMAX mode: interleaver parameters P1
15-9	Reserved	Reserved
8-0	INTVPAR0	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: Interleaver parameters f1 in formula y(i) = (f1*i + f2*i^2) mod K • WiMAX mode: interleaver parameters P0

Figure 4-40. ENC Local Header Word 9

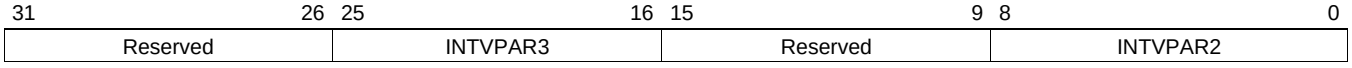


Table 4-42. ENC Local Header Word 9 Details

Bits	Field Name	Description
31-26	Reserved	Reserved
25-16	INTVPAR3	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: must be set to 0 • WiMAX mode: interleaver parameters P3
15-9	Reserved	Reserved
8-0	INTVPAR2	Interleaver generator parameter. • WCDMA mode: must be set to 0 • LTE mode: must be set to 0 • WiMAX mode: interleaver parameters P2

4.4.4 Error Bit Definitions

The ENC error bits are defined in Table 4-43. These are the errors bits that will be recorded in the ENC data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-43. ENC Error Bit Definitions

BIT	NAME
6	Detected one of the following: • WiMAX used with convolutional decoding • Invalid RADIO_STANDARD
5	Detected one of the following: • Code block size is outside valid range – Turbo decoding valid code block size range is 40 - 8192 – Convolutional decoding valid code block size range is 8 - 8192 • CRC/SCR enabled for WCDMA or LTE with a code block size that is not a multiple of 8 • WiMAX (with or without SCR/CRC programmed) with a code block which is not a multiple of 8
4	Invalid packet header configuration. Detected that NUM_CODE_BLKS_1 was equal to zero.
3	Invalid packet header configuration. Detected that BLOCK_SIZE_1 was equal to zero.
2	Detected late end-of-packet. This means that there was more input data than was expected based on the packet header parameters.
1	The amount of input data received did not match what was expected based on the packet header parameters.
0	Detected early end-of-packet. This means that there was less input data than was expected based on the packet header parameters.

4.5 RM

The RM submodule performs rate matching for all the standards supported by the BCP. For LTE, DL-SCH/PDSCH and PDCCH processing are supported and the input data for DL-SCH can be hard bits or soft bits (for uplink interference cancellation re-encoding). DL-DPDCH (Rel-99), DL-PDSCH (HSPA), and E-DCH FDD Turbo are also supported as well as WiMAX 802.16e. The RM submodule performs bit separation of TD-SCDMA Rel-99 Turbo encoded input data produced by the INT submodule and bit interleaving of convolutional or Turbo encoded input data from the ENC submodule. It also performs output data scrambling when needed for TDD modes. The input and output data formats as well as packet configuration fields differ by standard and are described in the following sections.

4.5.1 Data Format

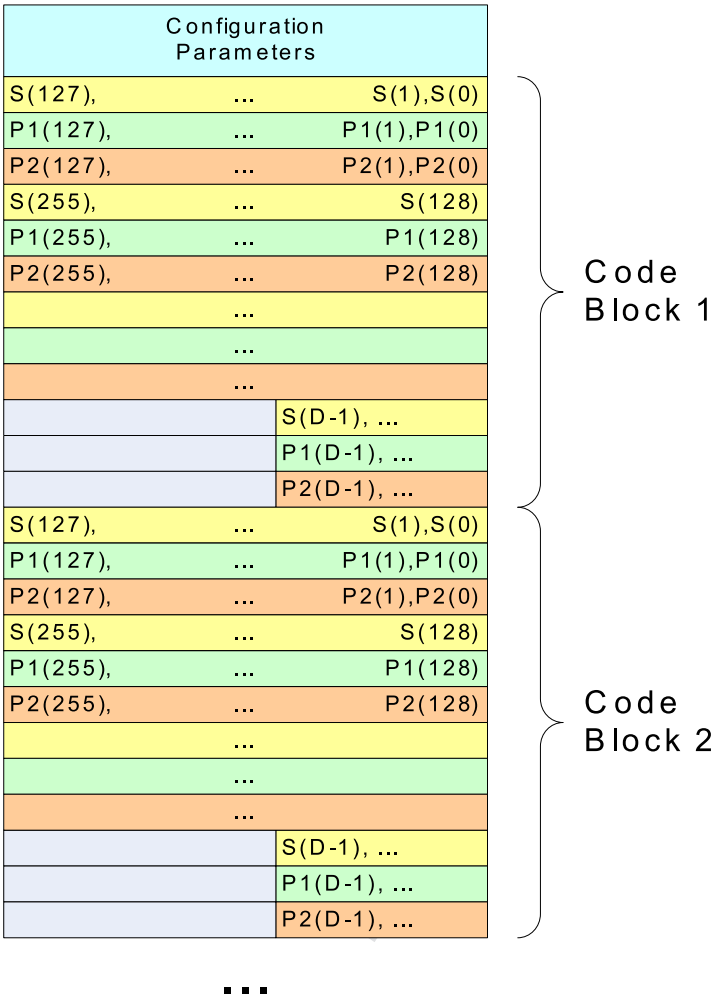
The input data to the RM submodule is received from the BCP streaming switch over the 128-bit streaming interface. The first data bit in time is located at the least significant bit position (bit 0) of 128-bit word.

4.5.1.1 LTE - Input Data

For LTE, the RM submodule takes in a single transport block from the streaming interface and outputs a single transport block to the streaming interface. The input transport block is separated into code blocks and the data processing that follows is performed on a code block basis. The input data can be either hard bits (usually from the ENC submodule for downlink) or soft bits (sent by the DSP and generated by the TCP3D for uplink interference cancellation).

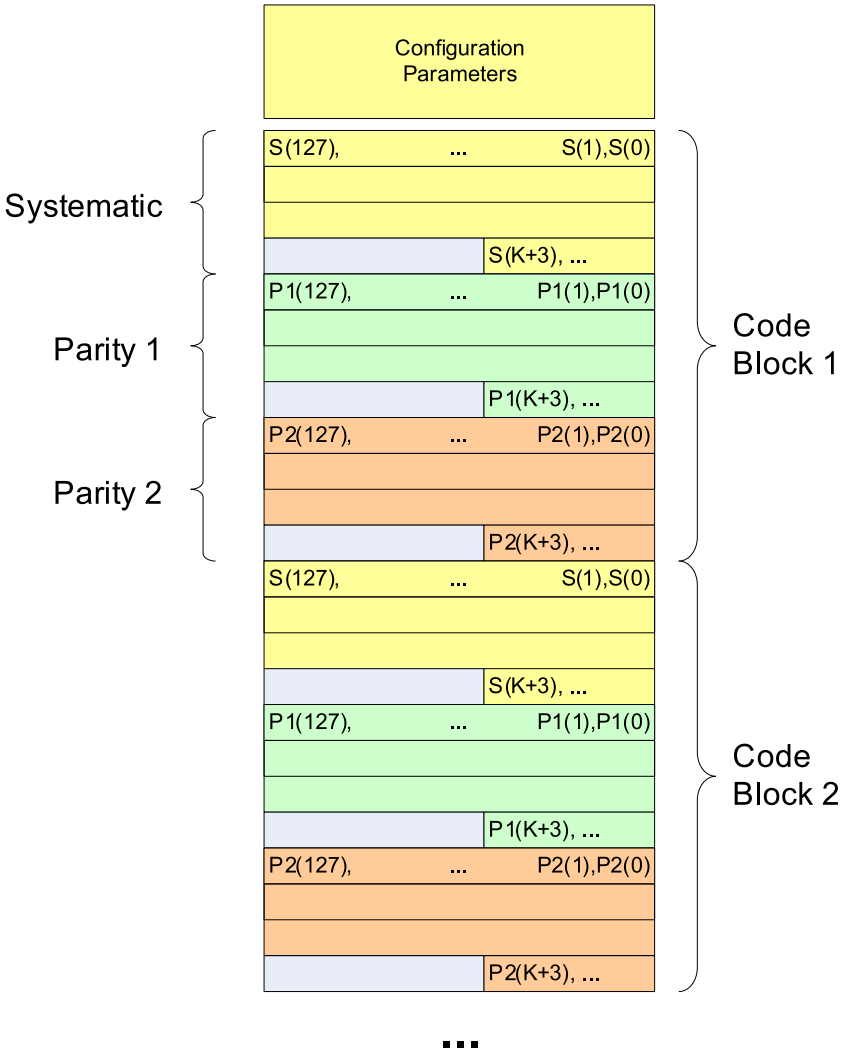
Each code block in the input transport block must start on a 128-bit boundary. The three components of the code block, systematic, parity 1 and parity 2 bits are transferred as shown in Figure 4-41. For LTE, the number of bits D = K+4 for DL-SCH and D = K for PDCCH. Although in the LTE PDCCH case there is no restriction in the RM submodule on the number of code blocks per packet, it is expected that in a typical scenario the BCP will process one code block per packet, since within the CRC submodule it can attach the scrambled CRC bits only to one code block per packet.

Figure 4-41. Hard Bits Format at the Input to Rate Matching Submodule in LTE Mode



Soft bits are packed as one soft bit per byte. They may be received from the DSP as one code block per packet. The traffic manager would then concatenate the code blocks into one packet using the “many-to-one” feature and pass them to RM submodule as one transport block per packet. The code block components (systematic, parity1 and parity2 bits) always start on 128-bit boundary as shown in Figure 4-42. The soft tail bits must be added by the DSP, as they are not produced by the TCP3D.

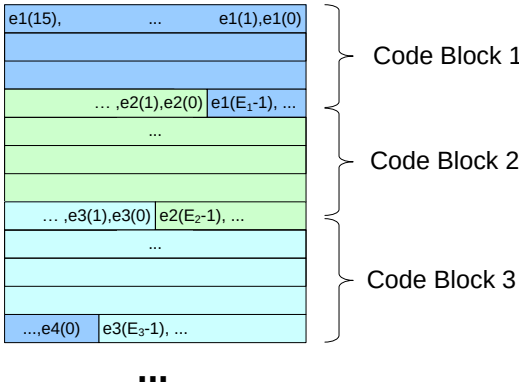
Figure 4-42. Soft Bits Format at the Input to Rate Matching Submodule



4.5.1.2 LTE - Output Data

The output bits are always soft bits (one per byte), even when the input bits are hard bits. The RM submodule is responsible for concatenating rate matched code blocks. There is no gap between the output rate matched code blocks. The output bits are generally sent to the MOD submodule which ignores the code block boundary. The output bit alignment is shown in Figure 4-43.

Figure 4-43. Bit Alignment at the Output of RM Submodule



When the input bits are hard bits, the bit translation is shown in Table 4-44.

Table 4-44. RM LTE Output Bit Translation for Hard Bit Formatted Code Blocks

Input data bit	Output data	
	LTE downlink	LTE PIC/SIC
0	0x00	0x7f (+127)
1	0x01	0x81 (-127)

4.5.1.3 WCDMA/TD-SCDMA - Input Data

The input packet to the RM module consists of one to six transport channels (TrCHs). The data is packed as hard bits. When multiple TrCHs exist in a packet, padding is inserted as needed, such that each TrCH begins on a 128-bit boundary. The local packet header for the RM block contains a length field for each TrCH. If a TrCH's length is 0, that TrCH is not present. For example, if TrCH 0 has a length of 0, TrCH 1 has a non-0 length, TrCH 2 has a 0 length, and TrCH 3 has a non-0 length, the TrCHs would come TrCH 1 first and TrCH 3 second. The largest TrCH size supported is 42,252 bits.

For the Turbo encoding in WCDMA Rel-99 and HSPA and TD-SCDMA HSPA, the input data comes from the ENC block and is received sequentially as 128-bits of systematic, 128-bits of parity 1, and 128 bits of parity 2 as three 128-bit words that will be bit interleaved inside the RM submodule before performing the rate-matching operation. The input format is shown in Table 4-45.

Table 4-45. Input Format for WCDMA Rel-99/HSPA and TD-SCDMA HSPA

Input format produced by ENC block (Turbo or 1/3-rate convolution)							
Word0	Sys[127]				…	Sys[1]	Sys[0]
Word1	Parity1[127]				…	Parity1[1]	Parity1[0]
Word2	Parity2[127]				…	Parity2[1]	Parity2[0]
Input format produced by ENC block (1/2-rate convolution)							
Word0	Output0[127]				…	Output0[1]	Output0[0]
Word1	Output1[127]				…	Output1[1]	Output1[0]

For TD-SCDMA Rel-99 Turbo encoding, the input data comes from the INT block and is already in bit interleaved order (systematic bit, parity1 bit, parity2 bit …) since the 1st interleaver is responsible for merging systematic parity 1 and parity 2 bits.

The RM output data is always packed into 128-bit words. It is transmitted starting in bit 0 of the first 128-bit word. TrCHs are packed into a single continuous stream with no padding between them.

4.5.1.4 WCDMA/TD-SCDMA - Output Data

The output data is packed as hard bits and ordered differently for Rel-99 (WCDMA and TD-SCDMA) and HSDPA (WCDMA and TD-SCDMA). The Rel-99 format is compatible with 3GPP 25.212 section 4.2.7.4.2 as used for Rel-99 and E-DCH channels. The output ordering is interleaved – systematic bit, parity 1 bit, parity 2 bit – with bits repeated or punctured as necessary. This is illustrated in the following example where puncturing is used. The output bits are labelled y(i1, i2, i3), where i1 defines whether the stream is systematic (i1 = 1), parity 1 (i1 = 2), or parity 2 (i1 = 3), i2 defines the transport block number, and i3 is the bit within the transport block. Output ordering is across in rows and then down. Table 4-46 shows the data marked (shaded) for puncturing, Table 4-47 shows the results after puncturing is completed, and Table 4-48 shows how the output data is packed into the 128-bit output bus.

Table 4-46. Output data in 32-bit words – punctured bits shaded

MSB															LSB
y1,1,6	y3,1,5	y2,1,5	y1,1,5	y3,1,4	y2,1,4	y1,1,4	y3,1,3	y2,1,3	y1,1,3	y3,1,2	y2,1,2	y1,1,2	y3,1,1	y2,1,1	y1,1,1
y2,1,11	y1,1,11	y3,1,10	y2,1,10	y1,1,10	y3,1,9	y2,1,9	y1,1,9	y3,1,8	y2,1,8	y1,1,8	y3,1,7	y2,1,7	y1,1,7	y3,1,6	y2,1,6
y3,1,16	y2,1,16	y1,1,16	y3,1,15	y2,1,15	y1,1,15	y3,1,14	y2,1,14	y1,1,14	y3,1,13	y2,1,13	y1,1,13	y3,1,12	y2,1,12	y1,1,12	y3,1,11
y1,1,22	y3,1,21	y2,1,21	y1,1,21	y3,1,20	y2,1,20	y1,1,20	y3,1,19	y2,1,19	y1,1,19	y3,1,18	y2,1,18	y1,1,18	y3,1,17	y2,1,17	y1,1,17

Table 4-47. Output data in 32-bit words – punctured bits removed

MSB															LSB
y1,1,10	y3,1,9	y2,1,8	y1,1,8	y3,1,7	y1,1,7	y2,1,6	y3,1,5	y1,1,5	y2,1,4	y1,1,4	y3,1,3	y2,1,2	y1,1,2	y3,1,1	y1,1,1
y1,1,20	y1,1,19	y2,1,18	y3,1,17	y1,1,17	y2,1,16	y1,1,16	y3,1,15	y2,1,14	y1,1,14	y3,1,13	y1,1,13	y2,1,12	y3,1,11	y1,1,11	y2,1,10
												…	y1,1,22	y3,1,21	y2,1,20

Table 4-48. Output data – packed into 128-bit output bus

MSB								LSB
bit127	….			bit4	bit3	bit2	bit1	bit0
	….	y1,1,5	y2,1,4	y1,1,4	y3,1,3	y2,1,2	y1,1,2	y3,1,1	y1,1,1
. . .								.

The HSDPA format is compatible with 3GPP 25.212 section 4.5.4.4 for HS-DSCH channels. The output data must be constructed into an HARQ table. The number of columns in this table is defined by the input parameter collect_cols and the number of rows by collect_rows. collect_rows will always be either 2, 4, or 6. collect_cols can be any value up to 7,200. The ordering is illustrated in the following example using puncturing. Table 4-49 shows the data marked (shaded) for puncturing. The systematic bits are output first followed by interleaved parity bits starting with parity2. Table 4-50 shows the results in the HARQ table format after puncturing is completed, where the thick line marks the boundary between the systematic bits and the interleaved parity bits. The output is down each column and then across the rows to form Table 4-51 showing how the output data is packed into the 128-bit output bus.

Table 4-49. Output data in 32-bit words – punctured bits marked in red

MSB															LSB
y1,1,16	y1,1,15	y1,1,14	y1,1,13	y1,1,12	y1,1,11	y1,1,10	y1,1,9	y1,1,8	y1,1,7	y1,1,6	y1,1,5	y1,1,4	y1,1,3	y1,1,2	y1,1,1
									…	y1,1,22	y1,1,21	y1,1,20	y1,1,19	y1,1,18	y1,1,17
y2,1,5	y3,1,5	y2,1,4	y3,1,4	y2,1,3	y3,1,3	y2,1,2	y3,1,2	y2,1,1	y3,1,1	…					
y2,1,13	y3,1,13	y2,1,12	y3,1,12	y2,1,11	y3,1,11	y2,1,10	y3,1,10	y2,1,9	y3,1,9	y2,1,8	y3,1,8	y2,1,7	y3,1,7	y2,1,6	y3,1,6
y2,1,21	y3,1,21	y2,1,20	y3,1,20	y2,1,19	y3,1,19	y2,1,18	y3,1,18	y2,1,17	y3,1,17	y2,1,16	y3,1,16	y2,1,15	y3,1,15	y2,1,14	y3,1,14
															…

Table 4-50. Output data in HARQ table format – punctured bits removed

HARQ	Col 1	Col 2	Col 3	Col 4	Col 5	Col6	Col7	…							
Row 1	y1,1,1	y1,1,5	y1,1,10	y1,1,14	y1,1,19										…
Row 2	y1,1,2	y1,1,7	y1,1,11	y1,1,16	y1,1,20										
Row 3	y1,1,4	y1,1,8	y1,1,13	y1,1,17	y1,1,22										
Row 4	y3,1,1	y2,1,4	y3,1,7	y2,1,10	y3,1,13	y2,1,16	y3,1,19								
Row 5	y2,1,2	y3,1,5	y2,1,8	y3,1,11	y2,1,14	y3,1,17	y2,1,20								
Row 6	y3,1,3	y2,1,6	y3,1,9	y2,1,12	y3,1,15	y2,1,18	y3,1,21								

Table 4-51. Output data – packed into 128-bit output bus

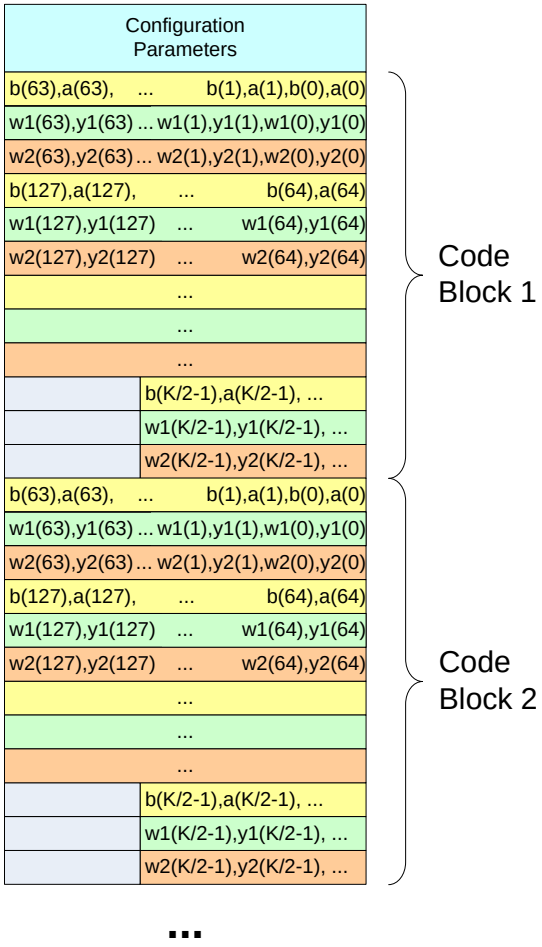
MSB												LSB
bit127	….							bit4	bit3	bit2	bit1	bit0
	….	y3,1,5	y2,1,4	y1,1,8	y1,1,7	y1,1,5	y3,1,3	y2,1,2	y3,1,1	y1,1,4	y1,1,2	y1,1,1
. . .												.

For Rel-99 convolutional coding or Turbo repeat, there is only a y(1,x,x) stream and therefore there is no bit interleaving or reordering.

4.5.1.5 WiMAX

In WiMAX mode, the RM submodule receives only hard bits, but is otherwise similar to LTE mode. The input data formatting is described in Figure 4-44. The output data is similar to LTE and formatted as soft bits.

Figure 4-44. Hard Bits Format at the Input to Rate Matching Submodule in WiMAX Mode



The bit translation is shown in Table 4-52.

Table 4-52. RM WiMAX Output Bit Translation

Input data bit	Output data
	WiMAX downlink
0	0x00
1	0x01

4.5.2 Memory Mapped Registers

The only memory mapped registers are the data logger registers and are described in the data logger section.

4.5.3 Packet Header Configuration Fields - LTE

4.5.3.1 LOCAL_HEADER_INFO (Word 0)

Figure 4-45. LTE Local Header Info Word (Word 0)

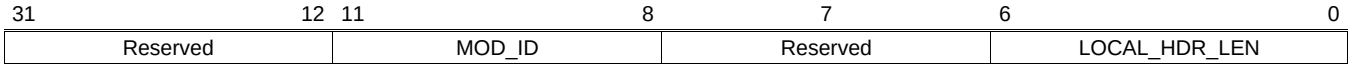


Table 4-53. LTE Local Header Info Word Details (Word 0)

Bit	Name	Description
31-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

4.5.3.2 MISC_CFG (Word 1)

Table 4-54. LTE Misc Config Word (Word 1)

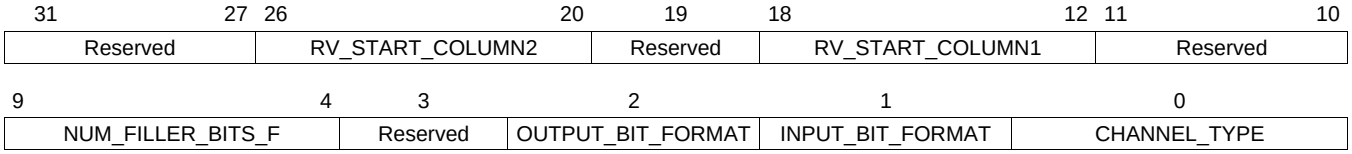


Table 4-55. LTE Misc Config Word Details (Word 1)

Bit	Name	Description
31-27	Reserved	Reserved
26-20	rv_start_column2	Value = 0-95 Starting permuted column during the dual matrix read phase for CBs of size K_2. • For LTE SCH: It is a function of the redundancy version index. The reading always starts from the top of the column (i.e., row0). Values >= 32 are in the P1/P2 matrix. • For LTE CCH: Values <32 in SYS matrix; Values from 32-63 are in the P1 matrix. Values >63 are in the P2 matrix. This value would normally be 0 for CCH.
19	Reserved	Reserved
18-12	rv_start_column1	Value = 0-95 Starting permuted column during the dual matrix read phase for CBs of size K_1. • For LTE SCH: It is a function of the redundancy version index. The reading always starts from the top of the column (i.e., row0). Values >= 32 are in the P1/P2 matrix. • For LTE CCH: Values <32 in SYS matrix; Values from 32-63 are in the P1 matrix. Values >63 are in the P2 matrix. This value would normally be 0 for CCH.
11-10	Reserved	Reserved
9-4	num_filler_bits_f	Value = 0-56 Number of filler bits (F) in the first code block. Set to 0 for CCH.
3	Reserved	Reserved
2	output_bit_format	Value = 0-1 Output bit format for hard bits: • 0 – “0” -> 0x00, “1” -> 0x01 • 1 – “0”-> 0x81, “1” -> 0x7F If input_bit_format =1, output_bit_format is ignored. Ignored for CCH.

Table 4-55. LTE Misc Config Word Details (Word 1) (continued)

Bit	Name	Description
1	input_bit_format	Value = 0-1 Input bit format: • 0 – Input bits are hard bits stored in packed format as one bit per stream bit, • 1 – Input bits are soft bits stored as one soft bit per stream byte. Set to 0 for CCH.
0	channel_type	Value = 0-1 LTE Channel type: • 0 – SCH • 1 – CCH

4.5.3.3 BUFFER_WRAP_OUT_CFG (Word 2)

Figure 4-46. LTE Buffer Wrap Output Config Word (Word 2)

31 22	21	20 16	15 6	5	4 0
PARAM_NCB2_ROW	Reserved	PARAM_NCB2 _COLUMN	PARAM_NCB1_ROW	Reserved	PARAM_NCB1_COLUMN

Table 4-56. LTE Buffer Wrap Output Config Word Details (Word 2)

Bit	Name	Description
31-22	param_ncb2_row	Value = 0-513 The row position of the last bit before the wrap for code blocks of length K2. RM interprets even values as P1 rows and odd values as P2 rows. Ignored for CCH since always wrap at end of buffer.
21	Reserved	Reserved
20-16	param_ncb2_column	Value = 0-31 The permuted column position of the last bit before the wrap for code blocks of length K2. RM always interprets this as being in the P1/P2 dual matrix. Ignored for CCH since always wrap at end of buffer.
15-6	param_ncb1_row	Value = 0-513 The row position of the last bit before the wrap for code blocks of length K1. RM interprets even values as P1 rows and odd values as P2 rows. Ignored for CCH since always wrap at end of buffer.
5	Reserved	Reserved
4-0	param_ncb1_column	Value = 0-31 The permuted column position of the last bit before the wrap for code blocks of length K1. RM always interprets this as being in the P1/P2 dual matrix. Ignored for CCH since always wrap at end of buffer.

4.5.3.4 CODE_BLK1_IN_CFG (Word 3)

Table 4-57. LTE Code Block 1 Input Config Word (Word 3)

31 30	29 16	15 6	5 0
Reserved	BLOCK_SIZE_K1	Reserved	NUM_CODE_BLOCKS_C1

Table 4-58. LTE Code Block 1 Input Config Word Details (Word 3)

Bit	Name	Description
31-30	Reserved	Reserved
29-16	block_size_k1	Value = 40-8192 Code block size (K_1), $K_r = K_1$ for r =0,…,C_1-1 prior to encoding. For SCH, RM adds 4 tail bits to this size internally. For CCH, use this value.
15-6	Reserved	Reserved
5-0	num_code_blocks_c1	Value = 0-63 Number of code blocks (C_1) of size K_1. Total number of code blocks in transport block is $C = C_1 + C_2$.

4.5.3.5 CODE_BLK1_OUT_CFG (Word 4)

Figure 4-47. LTE Code Block 1 Output Config Word (Word 4)

31 16	15 6	5 0
BLOCK_SIZE_E1	Reserved	NUM_CODE_BLOCKS_CE1

Table 4-59. LTE Code Block 1 Output Config Word Details (Word 4)

Bit	Name	Description
31-16	block_size_e1	Value = 0-65535 Output size (E_1), $E_r = E_1$, for r=0,…, C_{e1}-1.
15-6	Reserved	Reserved
5-0	num_code_blocks_ce1	Value = 0-63 Number of code blocks C_{e1} with output size E_1-. (Total number of code blocks in transport block is $C = C_{e1} + C_{e2}$).

4.5.3.6 CODE_BLK2_IN_CFG (Word 5)

Figure 4-48. LTE Code Block 2 Input Config Word (Word 5)

31 30	29 16	15 6	5 0
Reserved	BLOCK_SIZE_K2	Reserved	NUM_CODE_BLOCKS_C2

Table 4-60. LTE Code Block 2 Input Config Word Details (Word 5)

Bit	Name	Description
31-30	Reserved	Reserved
29-16	block_size_k2	Value = 40-8192 Code block size (K2), Kr = K2 for r =C1,…,(C1+ C2-1) prior to encoding. For SCH, RM adds 4 tail bits to this size internally. For CCH, this value is not modified internally.
15-6	Reserved	Reserved
5-0	num_code_blocks_c2	Value = 0-63 Number of code blocks (C_2) of size K_2. Total number of code blocks in transport block is $C = C_1 + C_2$.

4.5.3.7 CODE_BLK2_OUT_CFG (Word 6)

Figure 4-49. LTE Code Block 2 Output Config Word (Word 6)

31 16	15 6	5 0
BLOCK_SIZE_E2	Reserved	NUM_CODE_BLOCKS_CE2

Table 4-61. LTE Code Block 2 Output Config Word Details (Word 6)

Bit	Name	Description
31-16	block_size_e2	Value = 0-65535 Output size E_2, $E_r = E_2$, for r=C_{e1},…, (C_{e1}+ C_{e2}-1).
15-6	Reserved	Reserved
5-0	num_code_blocks_ce2	Value = 0-63 Number of code blocks C_{e2} with output size E_2.

4.5.4 Packet Header Configuration Fields - WCDMA/TD-SCDMA

Table 4-62. Packet Header Configuration Summary

Quad Word	Word3	Word2	Word1	Word0
0	sys0_len	Common	Common	Common
1	p0_par1_len	sys0_plus2 sys0_alpha sys0_beta sys0_puncture sys0_turbo	sys0_minus2	sys0_init2
2	p0_par1_init2	p0_par1_plus1	p0_par1_minus1	p0_par1_init1
3	p1_par1_init1	p1_par1_len	p1_par1_plus2	p0_par1_minus2
4	p1_par1_minus2	p1_par1_init2	p1_par1_plus1	p1_par1_minus1
5	sys1_alpha sys1_beta sys1_puncture sys1_turbo	sys1_minus2 sys1_plus2	sys1_len sys1_init2	p1_par1_plus2
6	sys3_len sys3_init2	sys2_alpha sys2_beta sys2_puncture sys2_turbo	sys2_minus2 sys2_plus2	sys2_len sys2_init2
7	sys4_minus2 sys4_plus2	sys4_len sys4_init2	sys3_alpha sys3_beta sys3_puncture sys3_turbo	sys3_minus2 sys3_plus2
8	sys5_alpha sys5_beta sys5_puncture sys5_turbo	sys5_minus2 sys5_plus2	sys5_len sys5_init2	sys4_alpha sys4_beta sys4_puncture sys4_turbo
9	p1_par2_minus2 p1_par2_plus2	p1_par2_len p1_par2_init2	p1_par1_minus2 p1_par1_plus2	p1_par1_len p1_par1_init2

Table 4-63. Packet Header Configuration Details

Word	Bits	Name	Description
0	5-0	local_hdr_len	Local header length in 32-bit words, excluding this word (Word 0).
	7-6	Reserved	Reserved for PPB usage
	11-8	mod_id	Module ID for this submodule (see Table 4-1 table)
	15-12	Reserved	Reserved for PPB usage
	16	input_order	Input Ordering: • 0 - Bit interleaved from Interleaver module • 1 - Quad word interleaved from Encoder module
	17	half_rate	Input encoding: • 0 -1/3 rate convolution or turbo encoded • 1 - 1/2 rate convolution encoded Note: This field is ignored when input_order = 0. When it is used (input_order = 1), all TrCHs must use the same rate.
	31-18	Reserved	Reserved for engine usage

Table 4-63. Packet Header Configuration Details (continued)

Word	Bits	Name	Description
1	23-0	collect_cols	Value - any Collect columns: Number of columns in bit collection table for the HSDPA turbo channel
	26-24	collect_rows	Value - any Collect rows: Number of rows in bit collection table for the HSDPA turbo channel
	31-27	Reserved	Reserved
2	15-0	num_scram	Value - any Scrambler initialization value for TD-SCDMA systematic HSDPA. RM performs scrambling for TDD HSDPA/HSUPA only. Set num_scram to 0x01000000 for scrambling specified in 3GPP 25.222 sections 4.5.5 (HSDPA) and 4.8.5 (HSUPA). Set to 0 for other standards.
	31-16	Reserved	Reserved
3	23-0	sys0_len	Value - any sys0 length (in bits) before rate-matching
	31-24	Reserved	Reserved
4	23-0	sys0_init2	Value - any sys0 initial value for 2nd rate-matching loop
	31-24	Reserved	Reserved
5	23-0	sys0_minus2	Value - any sys0 minus value for 2nd rate-matching loop
	31-24	Reserved	Reserved
6	23-0	sys0_plus2	Value - any sys0 plus value for 2nd rate-matching loop
	25-24	sys0_alpha	sys0 TD-SCDMA alpha value for bit separation: • 0 - Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 - Use α_{sys}, α_{p1}, α_{p2} =,0,2,1
	27-26	sys0_beta	Value - 0,1,2 sys0 TD-SCDMA beta value for bit separation
	28	sys0_puncture	Puncture flag: • 0 - Perform repeat in second rate matching loop • 1 - Perform puncture in second rate matching loop
	30-29	sys0_turbo	Value - 0,1,2,3 Indicates what type of channel this is: • 0 – Convolutional or Rel-99 Turbo Repeat • 1 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p0* parity parameters • 2 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p1* parity parameters • 3 – HSDPA style turbo channel use p0* parameters Note that if this channel is an HSDPA channel, all other channel lengths must be 0 (no mixing of HSDPA and non-HSDPA channels in the same packet)
	31	Reserved	Reserved
7	23-0	p0_par1_len	Value - any p0 Parity 1 length (in bits) before rate-matching
	31-24	Reserved	Reserved
8	23-0	p0_par1_init1	Value - any p0 Parity 1 initial value for 1st rate-matching loop
	31-24	Reserved	Reserved
9	23-0	p0_par1_minus1	Value - any p0 Parity 1 minus value for 1st rate-matching loop
	31-24	Reserved	Reserved

Table 4-63. Packet Header Configuration Details (continued)

Word	Bits	Name	Description
10	23-0	p0_par1_plus1	Value - any p0 Parity 1 plus value for 1st rate-matching loop
	31-24	Reserved	Reserved
11	23-0	p0_par1_init2	Value - any p0 Parity 1 initial value for 2nd rate-matching loop
	31-24	Reserved	Reserved
12	23-0	p0_par1_minus2	Value - any p0 Parity 1 minus value for 2nd rate-matching loop
	31-24	Reserved	Reserved
13	23-0	p0_par1_plus2	Value - any p0 Parity 1 plus value for 2nd rate-matching loop
	31-24	Reserved	Reserved
14	23-0	p0_par2_len	Value - any p0 Parity 2 length (in bits) before rate-matching
	31-24	Reserved	Reserved
15	23-0	p0_par2_init1	Value - any p0 Parity 2 initial value for 1st rate-matching loop
	31-24	Reserved	Reserved
16	23-0	p0_par2_minus1	p0 Parity 2 minus value for 1st rate-matching loop
	31-24	Reserved	Reserved
17	23-0	p0_par2_plus1	p0 Parity 2 plus value for 1st rate-matching loop
	31-24	Reserved	Reserved
18	23-0	p0_par2_init2	p0 Parity 2 initial value for 2nd rate-matching loop
	31-24	Reserved	Reserved
19	23-0	p0_par2_minus2	p0 Parity 2 minus value for 2nd rate-matching loop
	31-24	Reserved	Reserved
20	23-0	p0_par2_plus2	p0 Parity 2 plus value for 2nd rate-matching loop
	31-24	Reserved	Reserved
21	15-0	sys1_len	Channel 1 length (in bits) before rate-matching
	31-16	sys1_init2	Channel 1 initial value for 2nd rate-matching loop
22	15-0	sys1_minus2	Channel 1 minus value for 2nd rate-matching loop
	31-16	sys1_plus2	Channel 1 plus value for 2nd rate-matching loop
23	1-0	sys1_alpha	Channel 1 TD-SCDMA alpha value for bit separation: • 0 - Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 - Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
	3-2	sys1_beta	Channel 1TD-SCDMA beta value for bit separation
	4	sys1_puncture	Puncture flag: • 0 - Perform repeat in second rate matching loop • 1 - Perform puncture in second rate matching loop
	6-5	sys1_turbo	Value - 0,1,2 Indicates what type of channel this is: • 0 – Convolutional or Rel-99 Turbo Repeat • 1 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p0* parity parameters • 2 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p1* parity parameters • 3 – Reserved
	31-7	Reserved	Reserved
24	15-0	sys2_len	Value - any Channel 2 length (in bits) before rate-matching
	31-16	sys2_init2	Value - any Channel 2 initial value for 2nd rate-matching loop

Table 4-63. Packet Header Configuration Details (continued)

Word	Bits	Name	Description
25	15-0	sys2_minus2	Value - any Channel 2 minus value for 2nd rate-matching loop
	31-16	sys2_plus2	Value - any Channel 2 plus value for 2nd rate-matching loop
26	1-0	sys2_alpha	Channel 2 TD-SCDMA alpha value for bit separation: • 0 - Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 - Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
	3-2	sys2_beta	Value - 0,1,2 Channel 2TD-SCDMA beta value for bit separation
	4	sys2_puncture	Puncture flag: • 0 - Perform repeat in second rate matching loop • 1 - Perform puncture in second rate matching loop
	6-5	sys2_turbo	Value - 0,1,2 Indicates what type of channel this is: • 0 – Convolutional or Rel-99 Turbo Repeat • 1 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p0* parity parameters • 2 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p1* parity parameters • 3 – Reserved
	31-7	Reserved	Reserved
27	15-0	sys3_len	Value - any Channel 3 length (in bits) before rate-matching
	31-16	sys3_init2	Value - any Channel 3 initial value for 2nd rate-matching loop
28	15-0	sys3_minus2	Value - any Channel 3 minus value for 2nd rate-matching loop
	31-16	sys3_plus2	Value - any Channel 3 plus value for 2nd rate-matching loop
29	1-0	sys3_alpha	Channel 3 TD-SCDMA alpha value for bit separation: 0 - Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 1 - Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
	3-2	sys3_beta	Value - 0,1,2 Channel 3TD-SCDMA beta value for bit separation
	4	sys3_puncture	Puncture flag: 0 - Perform repeat in second rate matching loop 1 - Perform puncture in second rate matching loop
	6-5	sys3_turbo	Value - 0,1,2 Indicates what type of channel this is • 0 – Convolutional or Rel-99 Turbo Repeat • 1 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p0* parity parameters • 2 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p1* parity parameters • 3 – Reserved
	31-7	Reserved	Reserved
30	15-0	sys4_len	Value - any Channel 4 length (in bits) before rate-matching
	31-16	sys4_init2	Value - any Channel 4 initial value for 2nd rate-matching loop
31	15-0	sys4_minus2	Value - any Channel 4 minus value for 2nd rate-matching loop
	31-16	sys4_plus2	Value - any Channel 4 plus value for 2nd rate-matching loop

Table 4-63. Packet Header Configuration Details (continued)

Word	Bits	Name	Description
32	1-0	sys4_alpha	Channel 4 TD-SCDMA alpha value for bit separation: • 0 - Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 - Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
	3-2	sys4_beta	Value - 0,1,2 Channel 4 TD-SCDMA beta value for bit separation
	4	sys4_puncture	Puncture flag: • 0 - Perform repeat in second rate matching loop • 1 - Perform puncture in second rate matching loop
	6-5	sys4_turbo	Value - 0,1,2 Indicates what type of channel this is • 0 – Convolutional or Rel-99 Turbo Repeat • 1 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p0* parity parameters • 2 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p1* parity parameters • 3 – Reserved
	31-7	Reserved	Reserved
33	15-0	sys5_len	Value - any Channel 5 length (in bits) before rate-matching
	31-16	sys5_init2	Value - any Channel 5 initial value for 2nd rate-matching loop
34	15-0	sys5_minus2	Value - any Channel 5 minus value for 2nd rate-matching loop
	31-16	sys5_plus2	Value - any Channel 5 plus value for 2nd rate-matching loop
35	1-0	sys5_alpha	Channel 5 TD-SCDMA alpha value for bit separation: • 0 - Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 - Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
	3-2	sys5_beta	Value - 0,1,2 Channel 5 TD-SCDMA beta value for bit separation
	4	sys5_puncture	Puncture flag: • 0 - Perform repeat in second rate matching loop • 1 - Perform puncture in second rate matching loop
	6-5	sys5_turbo	Value - 0,1,2 Indicates what type of channel this is • 0 – Convolutional or Rel-99 Turbo Repeat • 1 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p0* parity parameters • 2 – Rel-99 style Turbo Puncture or E-DCH FDD Turbo, use p1* parity parameters • 3 - Reserved
	31-7	Reserved	Reserved
36	15-0	p1_par1_len	Value - any p1 Parity 1 length (in bits) before rate-matching
	31-16	p1_par1_init2	Value - any p1 Parity 1 initial value for 2nd rate-matching loop
37	15-0	p1_par1_minus2	Value - any p1 Parity 1 minus value for 2nd rate-matching loop
	31-16	p1_par1_plus2	Value - any p1 Parity 1 plus value for 2nd rate-matching loop
38	15-0	p1_par2_len	Value - any p1 Parity 2 length (in bits) before rate-matching
	31-16	p1_par2_init2	Value - any p1 Parity 2 initial value for 2nd rate-matching loop

Table 4-63. Packet Header Configuration Details (continued)

Word	Bits	Name	Description
39	15-0	p1_par2_minus2	Value - any p1 Parity 2 minus value for 2nd rate-matching loop
	31-16	p1_par2_plus2	Value - any p1 Parity 2 plus value for 2nd rate-matching loop

4.5.5 Packet Header Configuration Fields - WiMAX

4.5.5.1 LOCAL_HEADER_INFO (Word 0)

Figure 4-50. WiMAX Local Header Info Word

31 12	11 8	7	6 0
Reserved	MOD_ID	Reserved	LOCAL_HDR_LEN

Table 4-64. WiMAX Local Header Info Word Details

Bit	Name	Description
31-12	Reserved	Value - NA Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

4.5.5.2 CODE_BLK1_IN_CFG (Word 1)

Figure 4-51. WiMAX Code Block 1 Input Config Word (Word 1)

31 30	29 16	15 9	8 0
Reserved	BLOCK_SIZE_kK1	Reserved	NUM_CODE_BLOCKS_C1

Table 4-65. WiMAX Code Block 1 Input Config Word Details (Word 1)

Bit	Name	Description
31-30	Reserved	Value - NA
29-16	BLOCK_SIZE_kK1	Value - 48-8192 Code block size (K_1), $K_r = K_1$ for r = 0,…,C_1-1
15-9	Reserved	Value - NA
8-0	NUM_CODE_BLOCKS_C1	Value - 0-511 Number of code blocks (C_1) of size K_1. (Total number of CBs in burst is $C = C_1 + C_2 + C_3$.) This is also used as the number of code blocks C_{e1} with output size E_1.

4.5.5.3 CODE_BLK1_OUT_CFG (Word 2)

Figure 4-52. WiMAX Code Block 1 Output Config Word (Word 2)

31 16	15 7	6 4	3 0
BLOCK_SIZE_E1	Reserved	PARAM_J1	PARAM_M1

Table 4-66. WiMAX Code Block 1 Output Config Word Details (Word 2)

Bit	Name	Description
31-16	BLOCK_SIZE_E1	Value - 0-65535 Output size (E_1), $E_r = E_1$, for r = 0,…, C_1-1
15-7	Reserved	Value - NA
6-4	PARAM_J1	Value - ≤4 WiMAX sub-block interleaver parameter j for code block of size K_1
3-0	PARAM_M1	Value - ≤10 WiMAX sub-block interleaver parameter m for code blocks of size K_1

4.5.5.4 CODE_BLK2_IN_CFG (Word 3)

Figure 4-53. WiMAX Code Block 2 Input Config Word (Word 3)

31 30	29 16	15 2	1 0
Reserved	BLOCK_SIZE_K2	Reserved	NUM_CODE_BLOCKS_C2

Table 4-67. WiMAX Code Block 2 Input Config Word Details (Word 3)

Bit	Name	Description
31-30	Reserved	Value - NA
29-16	BLOCK_SIZE_K2	Value - 48-8192 Code block size (K_2), $K_r = K_2$ for $r = C_1$,.... $C_1 + C_{2-1}$
15-2	Reserved	Value - NA
1-0	NUM_CODE_BLOCKS_C2	Value 0-3 Number of code blocks (C_2) of size K_2. This is also used as the number of code blocks C_{e2} with output size E_2.

4.5.5.5 CODE_BLK2_OUT_CFG (Word 4)

Figure 4-54. WiMAX Code Block 2 Output Config Word (Word 4)

31 16	15 7	6 4	3 0
BLOCK_SIZE_E2	Reserved	PARAM_J2	PARAM_M2

Table 4-68. WiMAX Code Block 2 Output Config Word Details (Word 4)

Bit	Name	Description
31-16	BLOCK_SIZE_E2	Value - 0-65535 Output size (E_2), $E_r = E_2$, for $r = C1$,…, $C_1 + C_2$-1
15-7	Reserved	Value - NA
6-4	PARAM_J2	Value - ≤4 WiMAX sub-block interleaver parameter j for code block of size K_2
3-0	PARAM_M2	Value - ≤10 WiMAX sub-block interleaver parameter m for code blocks of size K_2

4.5.5.6 CODE_BLK3_IN_CFG (Word 5)

Figure 4-55. WiMAX Code Block 3 Input Config Word (Word 5)

31 30	29 16	15 2	1 0
Reserved	BLOCK_SIZE_K3	Reserved	NUM_CODE_BLOCKS_C3

Table 4-69. WiMAX Code Block 3 Input Config Word Details (Word 5)

Bit	Name	Description
31-30	Reserved	Value - NA
29-16	BLOCK_SIZE_K3	Value - 48-8192 Code block size (K_3), $K_r = K_3$ for $r = C_2$, ... $C_3 + C_{3-1}$
15-2	Reserved	Value - NA
1-0	NUM_CODE_BLOCKS_C3	Value - 0-3 Number of code blocks (C_3) of size K_3. This is also used as the number of code blocks C_{e3} with output size E_3.

4.5.5.7 CODE_BLK3_OUT_CFG (Word 6)

Figure 4-56. WiMAX Code Block 3 Output Config Word (Word 6)

31 16	15 7	6 4	3 0
BLOCK_SIZE_E3	Reserved	PARAM_J3	PARAM_M3

Table 4-70. WiMAX Code Block 3 Output Config Word Details (Word 6)

Bit	Name	Description
31-16	BLOCK_SIZE_E3	Value - 0-65535 Output size (E_3), $E_r = E_3$, for $r = C_2,\ldots, C_2 + C_3 - 1$
15-7	Reserved	Value - NA
6-4	PARAM_J3	Value - ≤4 WiMAX sub-block interleaver parameter j for code block of size K_3
3-0	PARAM_M3	Value - ≤10 WiMAX sub-block interleaver parameter m for code blocks of size K_3

4.5.6 Error Bit Definitions

The RM error bits are defined in Table 4-71. These are the errors bits that will be recorded in the RM data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-71. RM Error Bit Definitions

BIT	NAME
6	Reserved
5	Reserved
4	Reserved
3	Reserved
2	Invalid packet header configuration. Detected at least one of the following: • block_size > 8192 • block_size_k1 = 0 • num_code_blocks_c1 = 0 • block_size_e1 = 0 • num_code_blocks_ce1 = 0 • num_code_blocks_c1 + num_code_blocks_c2 != num_code_blocks_ce1 + num_code_blocks_ce2 • param_ncb1_row out of range • param_ncb2_row out of range • invalid RADIO_STANDARD
1	Detected that there was more input data than was expected based on the packet header parameters.
0	Detected that there was less input data than was expected based on the packet header parameters.

4.6 MOD

The MOD submodule combines the functionalities of a scrambler, soft or hard modulator, and interleaver. This submodule is intended primarily for LTE, although the hard modulator can be used for WiMAX as well (in which case the scrambling operation is bypassed). It has two modes of operation. In the “soft modulator” mode (typically used for LTE PIC/SIC reconstruction of uplink data), the scrambler, soft modulator and interleaver are combined. In the “hard modulator” mode (typically used for LTE downlink), the scrambler and hard modulator are combined. In either mode, the scrambler operation can be disabled. The MOD submodule also performs CQI and RI insertion and ACK /NACK overwrite (both using dummy values). The case when a packet carries CQI data only (no payload information data) will be handled by the DSP and will not be sent for processing by the MOD soft modulator.

4.6.1 Data Format

4.6.1.1 Input Data

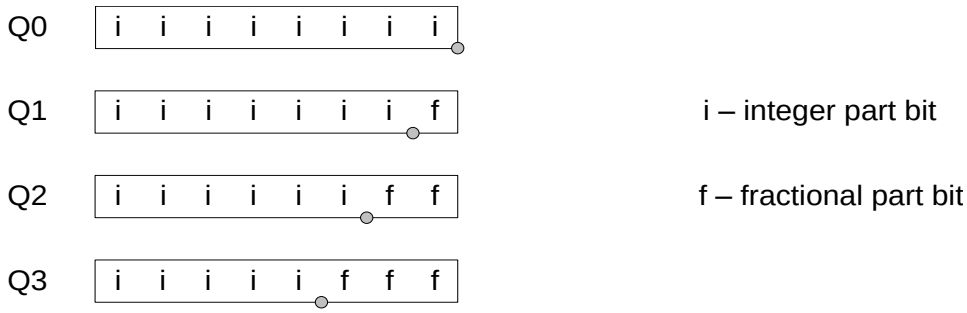
The MOD input data is usually provided by the RM submodule. The data is formatted as a stream of 8-bit integers. Each 8-bit integer will represent one of the following:

- Soft bit LLR in 2's complement format - used in the soft-decision PIC/SIC uplink LTE receiver
- Soft bit LLR in 2's complement format (extreme values: 0x7f for 1 and 0x81 for 0) - used in the hard-decision PIC/SIC uplink LTE receiver
- Bit to be hard modulated format (0x01 for 1 and 0x00 for 0) - used in the downlink LTE or WiMAX transmitter

Bits, represented by 8-bit integers (both the soft bits and the hard bits), are processed in groups of Qm, where Qm is the number of bits used to represent a constellation point (Qm = 1, 2, 4, 6, 8 for BPSK, QPSK, 16-QAM, 64-QAM, 256-QAM, respectively).

The precision of the input soft bits in “soft modulator” mode is programmable on a per packet basis via the q_format field of the MODE_SEL_CFG word in the packet configuration header. The available Q formats are shown in Figure 4-57.

Figure 4-57. MOD Input Soft Bit Precision



4.6.1.2 Output Data

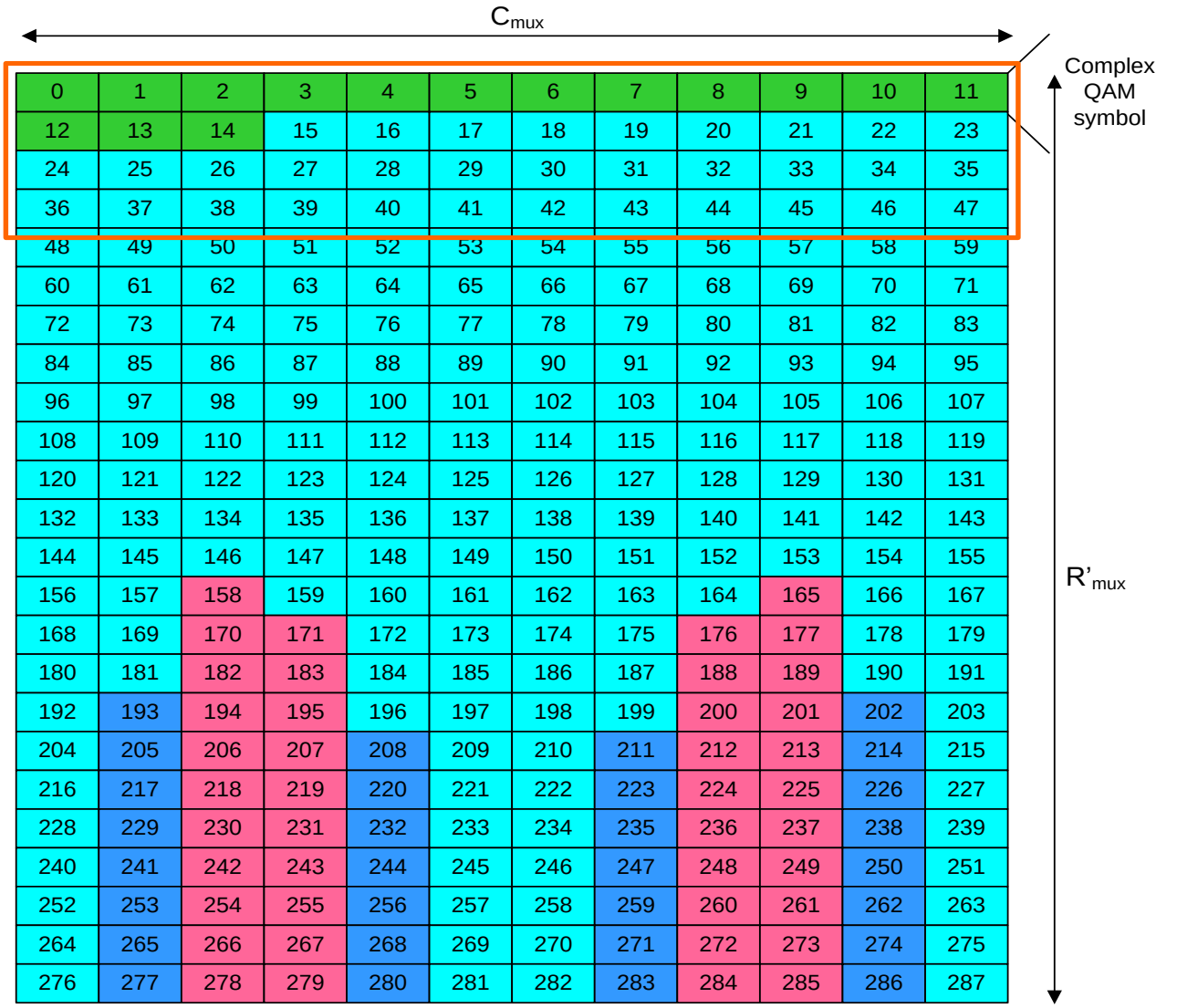
4.6.1.2.1 Soft Modulation Mode

The MOD output data format depends on the mode of operation. For the same reasons as listed in the SSL LTE input data section (Section 4.7.1.1), when MOD soft modulation mode is used, the typical sequence of operations from the 3GPP spec is modified to be as follows:

1. Scrambling (soft bits) in MOD
2. Soft modulation in MOD
3. Channel interleaving (stage 1) in MOD
4. Channel interleaving (stage 2) using EDMA

After scrambling and soft modulation, the data inside the MOD submodule can be represented as shown in the example in Figure 4-58. Dummy values are set to the most negative value (-127).

Figure 4-58. Example MOD Internal Data After Scrambling and Soft Modulation



The data is then interleaved as shown in Figure 4-59.

Figure 4-59. Example MOD "Soft Modulation" Mode Internal Data After Interleaving

R'_{mux}

C_{mux}																								
0	12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240	252	264	276
1	13	25	37	49	61	73	85	97	109	121	133	145	157	169	181	193	205	217	229	241	253	265	277
2	14	26	38	50	62	74	86	98	110	122	134	146	158	170	182	194	206	218	230	242	254	266	278
3	15	27	39	51	63	75	87	99	111	123	135	147	159	171	183	195	207	219	231	243	255	267	279
4	16	28	40	52	64	76	88	100	112	124	136	148	160	172	184	196	208	220	232	244	256	268	280
5	17	29	41	53	65	77	89	101	113	125	137	149	161	173	185	197	209	221	233	245	257	269	281
6	18	30	42	54	66	78	90	102	114	126	138	150	162	174	186	198	210	222	234	246	258	270	282
7	19	31	43	55	67	79	91	103	115	127	139	151	163	175	187	199	211	223	235	247	259	271	283
8	20	32	44	56	68	80	92	104	116	128	140	152	164	176	188	200	212	224	236	248	260	272	284
9	21	33	45	57	69	81	93	105	117	129	141	153	165	177	189	201	213	225	237	249	261	273	285
10	22	34	46	58	70	82	94	106	118	130	142	154	166	178	190	202	214	226	238	250	262	274	286
11	23	35	47	59	71	83	95	107	119	131	143	155	167	179	191	203	215	227	239	251	263	275	287

The data in the red box is then output by the BCP as shown in Figure 4-60.

Figure 4-60. Example MOD "Soft Modulation" Mode Output Data

127 96	95 64	63 32	31 0
I_{21} 36	Q_{21} 36	I_9 24	Q_9 24	I_{CQI} dummy12	Q_{CQI} dummy12	I_{CQI} dummy0	Q_{CQI} dummy0
I_{22} 37	Q_{22} 37	I_{10} 25	Q_{10} 25	I_{CQI} dummy13	Q_{CQI} dummy13	I_{CQI} dummy1	Q_{CQI} dummy1
I_{23} 38	Q_{23} 38	I_{11} 26	Q_{11} 26	I_{CQI} dummy14	Q_{CQI} dummy14	I_{CQI} dummy2	Q_{CQI} dummy2
I_{24} 39	Q_{24} 39	I_{12} 27	Q_{12} 27	I_0 15	Q_0 15	I_{CQI} dummy3	Q_{CQI} dummy3
I_{25} 40	Q_{25} 40	I_{13} 28	Q_{13} 28	I_1 16	Q_1 16	I_{CQI} dummy4	Q_{CQI} dummy4
I_{26} 41	Q_{26} 41	I_{14} 29	Q_{14} 29	I_2 17	Q_2 17	I_{CQI} dummy5	Q_{CQI} dummy5
I_{27} 42	Q_{27} 42	I_{15} 30	Q_{15} 30	I_3 18	Q_3 18	I_{CQI} dummy6	Q_{CQI} dummy6
I_{28} 43	Q_{28} 43	I_{16} 31	Q_{16} 31	I_4 19	Q_4 19	I_{CQI} dummy7	Q_{CQI} dummy7
I_{29} 44	Q_{29} 44	I_{17} 32	Q_{17} 32	I_5 20	Q_5 20	I_{CQI} dummy8	Q_{CQI} dummy8
I_{30} 45	Q_{30} 45	I_{18} 33	Q_{18} 33	I_6 21	Q_6 21	I_{CQI} dummy9	Q_{CQI} dummy9
I_{31} 46	Q_{31} 46	I_{19} 34	Q_{19} 34	I_7 22	Q_7 22	I_{CQI} dummy10	Q_{CQI} dummy10
I_{32} 47	Q_{32} 47	I_{20} 35	Q_{20} 35	I_8 23	Q_8 23	I_{CQI} dummy11	Q_{CQI} dummy11
.							
.							
.							

After all the data is output to memory, the EDMA must re-organize it slightly to complete the interleaving operation. When two layers are used, the internal data after interleaving would be as shown in Figure 4-61. The first layer (layer 0) is shown in darker colors. The layer mapping in the uplink is based on the principals of already defined layer mapping for the SU-MIMO case in the downlink. These changes related to the 2-layer split of a transport block have no effect on the operation of the soft modulator sub-module except for the variance computation.

Figure 4-61. Example MOD "Soft Modulation" Mode Internal Data After Interleaving (2-layer)

R'_{mux} →

C_{mux} ↓

0	12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240	252	264	276
1	13	25	37	49	61	73	85	97	109	121	133	145	157	169	181	193	205	217	229	241	253	265	277
2	14	26	38	50	62	74	86	98	110	122	134	146	158	170	182	194	206	218	230	242	254	266	278
3	15	27	39	51	63	75	87	99	111	123	135	147	159	171	183	195	207	219	231	243	255	267	279
4	16	28	40	52	64	76	88	100	112	124	136	148	160	172	184	196	208	220	232	244	256	268	280
5	17	29	41	53	65	77	89	101	113	125	137	149	161	173	185	197	209	221	233	245	257	269	281
6	18	30	42	54	66	78	90	102	114	126	138	150	162	174	186	198	210	222	234	246	258	270	282
7	19	31	43	55	67	79	91	103	115	127	139	151	163	175	187	199	211	223	235	247	259	271	283
8	20	32	44	56	68	80	92	104	116	128	140	152	164	176	188	200	212	224	236	248	260	272	284
9	21	33	45	57	69	81	93	105	117	129	141	153	165	177	189	201	213	225	237	249	261	273	285
10	22	34	46	58	70	82	94	106	118	130	142	154	166	178	190	202	214	226	238	250	262	274	286
11	23	35	47	59	71	83	95	107	119	131	143	155	167	179	191	203	215	227	239	251	263	275	287

In soft modulation mode, the output data consists of the following:

- Stream of 2-dimensional constellation points, each of which is represented by a pair of 16-bit integers; 16-bit for Inphase (real) and 16-bit for the Quadrature (imaginary). Four of these pairs are output at a time in order to fill completely the 128-bit payload. The I/Q order can be reversed via the "jack_bit" packet header configuration field.
- Cmux number of accumulated variance values (Cmux may take values 9, 10, 11, 12). Each of the accumulated variances is represented by a 32-bit word in IEEE 754 single precision floating point format. In the case of SU-MIMO (Single User Multiple Input Multiple Output) there may be two sets of Cmux values of accumulated variance. Accumulated variances are sent out at the end of the output payload.

The alignment of the MOD output in soft modulation mode is as shown in Figure 4-62.

Figure 4-62. MOD Output in Soft Modulation ModSPRUGZ1_tbl_145e

Soft Symbols

127 96	95 64	63 32	31 0
I_3 Q_3	I_2 Q_2	I_1 Q_1	I_0 Q_0
I_7 Q_7	I_6 Q_6	I_5 Q_5	I_4 Q_4

.
.
.

Accumulated Variances (normal operation)

127 96	95 64	63 32	31 0
Var_3	Var_2	Var_1	Var_0
Var_7	Var_6	Var_5	Var_4
Var_{11} or Reserved	Var_{10} or Reserved	Var_9 or Reserved	Var_8

Accumulated Variances (transport block split into two layers)

127 96	95 64	63 32	31 0
$Var_{0,3}$	$Var_{0,2}$	$Var_{0,2}$	$Var_{0,0}$
$Var_{0,7}$	$Var_{0,6}$	$Var_{0,5}$	$Var_{0,4}$
$Var_{0,11}$ or Reserved	$Var_{0,10}$ or Reserved	$Var_{0,9}$ or Reserved	$Var_{0,8}$
$Var_{1,3}$	$Var_{1,2}$	$Var_{1,1}$	$Var_{1,0}$
$Var_{1,7}$	$Var_{1,6}$	$Var_{1,5}$	$Var_{1,4}$
$Var_{1,11}$ or Reserved	$Var_{1,10}$ or Reserved	$Var_{1,9}$ or Reserved	$Var_{1,8}$

NOTE: Reserved values may not be all-0.

The MOD submodule outputs several (Cmux) accumulated values per transport block. Each of the values is of the form:

$$Var_j = \sum_{l=0}^{L_{Ldata}-1} \left(E_C\left\{X_{j,l}^2\right\} - \left|E_C\left\{X_{j,l}\right\}\right|^2 \right), j = 0,1,...C_{mux} - 1 \quad (1)$$

The accumulation is performed only on data carrying subcarriers. Subcarriers corresponding to CQI, RI and ACK/NACK are skipped in the process. Figure 4-63 illustrates which subcarriers are used in this computation (by an example).

Figure 4-63. Accumulated Variance Computation (1-layer case)

0	12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240	252	264	276
													Var0										
1	13	25	37	49	61	73	85	97	109	121	133	145	157	169	181	193	205	217	229	241	253	265	277
									Var1														
2	14	26	38	50	62	74	86	98	110	122	134	146	158	170	182	194	206	218	230	242	254	266	278
							Var2																
3	15	27	39	51	63	75	87	99	111	123	135	147	159	171	183	195	207	219	231	243	255	267	279
								Var3															
4	16	28	40	52	64	76	88	100	112	124	136	148	160	172	184	196	208	220	232	244	256	268	280
									Var4														
5	17	29	41	53	65	77	89	101	113	125	137	149	161	173	185	197	209	221	233	245	257	269	281
												Var5											
6	18	30	42	54	66	78	90	102	114	126	138	150	162	174	186	198	210	222	234	246	258	270	282
												Var6											
7	19	31	43	55	67	79	91	103	115	127	139	151	163	175	187	199	211	223	235	247	259	271	283
									Var7														
8	20	32	44	56	68	80	92	104	116	128	140	152	164	176	188	200	212	224	236	248	260	272	284
							Var8																
9	21	33	45	57	69	81	93	105	117	129	141	153	165	177	189	201	213	225	237	249	261	273	285
							Var9																
10	22	34	46	58	70	82	94	106	118	130	142	154	166	178	190	202	214	226	238	250	262	274	286
								Var10															
11	23	35	47	59	71	83	95	107	119	131	143	155	167	179	191	203	215	227	239	251	263	275	287
												Var11											

Accumulation of computed 21-bit variances is done over relevant carriers in a given SC-FDMA symbol. The accumulated value is a 32-bit integer, which can accommodate maximum of 110×12 values that may participate in the summation. After the whole transport block has been processed, all of those Cmux values are made available to be transferred to the DSP as 32-bit values representing the accumulated variance in the IEEE 754 base 2 floating-point format. When the CQI, RI and ACK/NACK symbols are inserted as groups of LLRs with saturated value (e.g., -127) the corresponding computed variance would be 0. Therefore, the summation may include those symbols as well as their contribution to the final accumulated variance is 0.

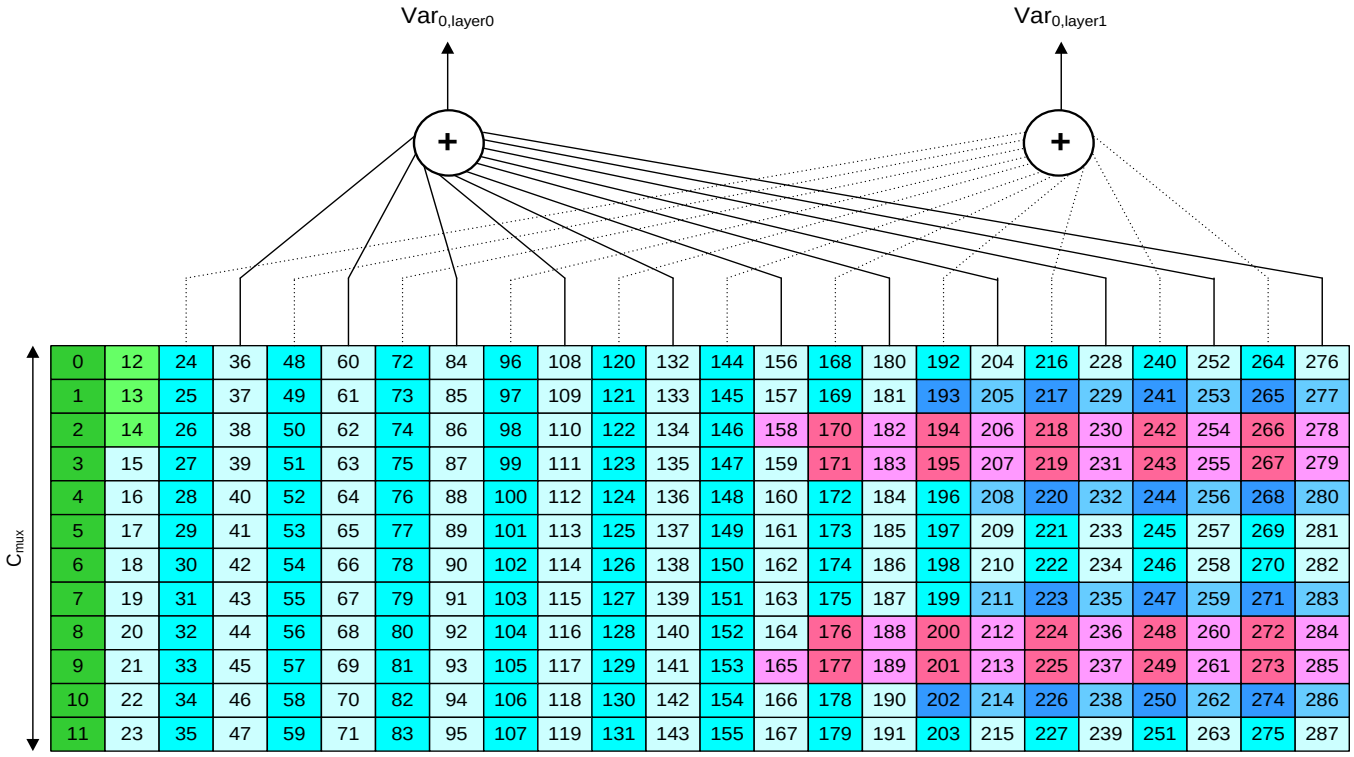
In the case when the transmit block is mapped into two layers, two sets of accumulated variances need to be computed (each set with Cmux elements). The computation is now performed as follows:

$$Var_{j,layer0} \overset{def}{=} Var_{0,j} = \sum_{l=0,(evencolumns)}^{L_{Ldata}-1} \left(E_C\left\{X_{j,l}^2\right\} - \left|E_C\left\{X_{j,l}\right\}\right|^2 \right), j = 0,1,...,C_{mux}-1$$

$$Var_{j,layer1} \overset{def}{=} Var_{1,j} = \sum_{l=0,(oddcolumns)}^{L_{Ldata}-1} \left(E_C\left\{X_{j,l}^2\right\} - \left|E_C\left\{X_{j,l}\right\}\right|^2 \right), j = 0,1,...,C_{mux}-1 \quad (2)$$

Figure 4-64 illustrates computation of $Var_{0,\ layer0}$ and $Var_{0,\ layer1}$. Only subcarriers in row 0 are used for $Var_{0,\ layer0}$ and $Var_{0,\ layer1}$ computations.

Figure 4-64. Accumulated Variance Computation (2-layer case)



4.6.1.2.2 Hard Modulation Mode

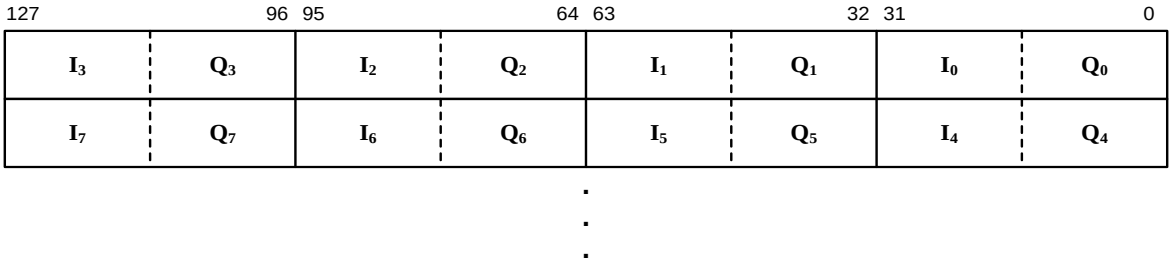
In hard modulation mode, the output data format depends on the selection of compressed or uncompressed output mode. The data consists of only one of the following.

- Stream of 2-dimensional constellation points, each of which is represented by a pair of 16-bit integers; 16-bit for Inphase (real) and 16-bit for the Quadrature (imaginary). Four of these pairs are output at a time in order to fill completely the 128-bit payload. This is refer red to as MOD output “uncompressed mode”. The I/Q order can be reversed via the “jack_bit” packet header configuration field.
- Stream of 2-dimensional constellation points, each of which is represented by an 8-bit integer, which in turn represents the index into a predefined table containing I and Q components of the constellation points. This is refer red to as a MOD output “compressed mode”. Only BPSK, QPSK, 16-QAM and 64-QAM are supported in the “compressed mode”.

The alignment of the MOD output in hard modulation mode is as shown in Figure 4-62.

Figure 4-65. MOD Output in Hard Modulation Mode

Hard Symbols (Uncompressed)



Hard Symbols (Compressed)

A lookup table containing the pre-scaled in-phase and quadrature components corresponding to each possible MOD “compressed mode” output index may be maintained by the DSP in DSP memory. The output indices assume a table of the structure shown in the following example tables for LTE and WiMAX with unscaled entries (prior to multiplying with the customer defined “unit” value).

Figure 4-66. Example of Non-scaled MOD Output "Compressed Mode" Index Lookup Table in DSP Memory

LTE

Index	Bits	I (Re)	Q (Im)	
0	0	1	1	BPSK
1	1	-1	-1	
2	00	1	1	QPSK
3	01	1	-1	
4	10	-1	1	
5	11	-1	-1	
6	0000	1	1	16-QAM
7	0001	1	3	
8	0010	3	1	
9	0011	3	3	
10	0100	1	-1	
11	0101	1	-3	
12	0110	3	-1	
13	0111	3	-3	
14	1000	-1	1	
15	1001	-1	3	
16	1010	-3	1	
17	1011	-3	3	
18	1100	-1	-1	
19	1101	-1	-3	
20	1110	-3	-1	
21	1111	-3	-3	
22	000000	3	3	64-QAM
23	000001	3	1	
24	000010	1	3	
25	000011	1	1	
26	000100	3	5	
27	000101	3	7	
28	000110	1	5	
29	000111	1	7	
30	001000	5	3	
31	001001	5	1	
32	001010	7	3	
33	001011	7	1	
34	001100	5	5	
35	001101	5	7	
36	001110	7	5	
37	001111	7	7	
38	010000	3	-3	
39	010001	3	-1	
40	010010	1	-3	
41	010011	1	-1	
42	010100	3	-5	
43	010101	3	-7	
44	010110	1	-5	
45	010111	1	-7	
46	011000	5	-3	
47	011001	5	-1	
48	011010	7	-3	
49	011011	7	-1	
50	011100	5	-5	
51	011101	5	-7	
52	011110	7	-5	
53	011111	7	-7	
54	100000	-3	3	
55	100001	-3	1	
56	100010	-1	3	
57	100011	-1	1	
58	100100	-3	5	
59	100101	-3	7	
60	100110	-1	5	
61	100111	-1	7	
62	101000	-5	3	
63	101001	-5	1	
64	101010	-7	3	
65	101011	-7	1	
66	101100	-5	5	
67	101101	-5	7	
68	101110	-7	5	
69	101111	-7	7	
70	110000	-3	-3	
71	110001	-3	-1	
72	110010	-1	-3	
73	110011	-1	-1	
74	110100	-3	-5	
75	110101	-3	-7	
76	110110	-1	-5	
77	110111	-1	-7	
78	111000	-5	-3	
79	111001	-5	-1	
80	111010	-7	-3	
81	111011	-7	-1	
82	111100	-5	-5	
83	111101	-5	-7	
84	111110	-7	-5	
85	111111	-7	-7	

WiMAX

Index	Bits	I (Re)	Q (Im)
0	0	1	0
1	1	-1	0
2	00	1	1
3	01	1	-1
4	10	-1	1
5	11	-1	-1
6	0000	1	1
7	0001	1	3
8	0010	1	-1
9	0011	1	-3
10	0100	3	1
11	0101	3	3
12	0110	3	-1
13	0111	3	-3
14	1000	-1	1
15	1001	-1	3
16	1010	-1	-1
17	1011	-1	-3
18	1100	-3	1
19	1101	-3	3
20	1110	-3	-1
21	1111	-3	-3
22	000000	3	3
23	000001	3	1
24	000010	3	5
25	000011	3	7
26	000100	3	-3
27	000101	3	-1
28	000110	3	-5
29	000111	3	-7
30	001000	1	3
31	001001	1	1
32	001010	1	5
33	001011	1	7
34	001100	1	-3
35	001101	1	-1
36	001110	1	-5
37	001111	1	-7
38	010000	5	3
39	010001	5	1
40	010010	5	5
41	010011	5	7
42	010100	5	-3
43	010101	5	-1
44	010110	5	-5
45	010111	5	-7
46	011000	7	3
47	011001	7	1
48	011010	7	5
49	011011	7	7
50	011100	7	-3
51	011101	7	-1
52	011110	7	-5
53	011111	7	-7
54	100000	-3	3
55	100001	-3	1
56	100010	-3	5
57	100011	-3	7
58	100100	-3	-3
59	100101	-3	-1
60	100110	-3	-5
61	100111	-3	-7
62	101000	-1	3
63	101001	-1	1
64	101010	-1	5
65	101011	-1	7
66	101100	-1	-3
67	101101	-1	-1
68	101110	-1	-5
69	101111	-1	-7
70	110000	-5	3
71	110001	-5	1
72	110010	-5	5
73	110011	-5	7
74	110100	-5	-3
75	110101	-5	-1
76	110110	-5	-5
77	110111	-5	-7
78	111000	-7	3
79	111001	-7	1
80	111010	-7	5
81	111011	-7	7
82	111100	-7	-3
83	111101	-7	-1
84	111110	-7	-5
85	111111	-7	-7

4.6.2 Memory Mapped Registers

The only memory mapped registers are the data logger registers and are described in the data logger section.

4.6.3 Packet Header Configuration Fields

4.6.3.1 UVA_VAL_CFG (Word 0)

Figure 4-67. Unit Value a Config (Word 0)

31 16	15 12	11 8	7 6	0
UVA_VAL	Reserved	MOD_ID	Reserved	LOCAL_HDR_LEN

Table 4-72. Unit Value a Config Details (Word 0)

Bit	Name	Description
31-16	UVA_VAL	Unit value ‘a’ for both soft and hard modulation. Applied by multiplying the standard constellation points (like those shown in Figure 4-58) by this value. This parameter is not used for hard “compressed-mode” output since the output is just an index in that mode.
15-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

4.6.3.2 MODE_SEL_CFG (Word 1)

Figure 4-68. Mode Select Config (Word 1)

31 25	24	23	22 16	15 12	11 10
Reserved	JACK_BIT	Reserved	B_TABLE_INDEX	Reserved	Q_FORMAT

9 8	7 4	3	2	1 0
CMUX_LN	MOD_TYPE_SEL	SCR_EN	SPLIT_MODE_EN	SH_MOD_SEL

Table 4-73. Mode Select Config Details (Word 1)

Bit	Name	Description
31-25	Reserved	Value - NA
24	jack_bit	IQ Output Reverse: • 0 = Output Q as lsb then I • 1 = Output I as lsb then Q
23	Reserved	Value - NA
22-16	b_table_index	Index into ‘B’ tables of initial vectors to create the ‘B’ matrices (soft modulation mode only). Please see Example 4-1 for an example code that can be used to calculate this parameter.
15-12	Reserved	Value - NA
11-10	q_format	LLR Q Format (soft modulation mode only): • 0 = Q0 • 1 = Q1 • 2 = Q2 • 3 = Q3
9-8	cmux_ln	C_{mux} length select (soft modulation mode only): • 0 = 9 Columns (Extended w/ SRS) • 1 = 10 Columns (Extended w/o SRS) • 2 = 11 Columns (Normal w/ SRS) • 3 = 12 Columns (Normal w/o SRS) Note: This field is ignored in Hard Modulation Mode.

Table 4-73. Mode Select Config Details (Word 1) (continued)

Bit	Name	Description
7-4	mod_type_sel	Modulation Type Select: • 0 = BPSK • 1 = QPSK • 2 = 16 QAM • 3 = 64 QAM • 4 = 256 QAM • 5-7 = n/a
3	scr_en	Scrambling Enable (for both hard and soft modulation LTE): • 0 = Scrambling Disabled (i.e., Bypassed) • 1 = Scrambling Enabled
2	split_mode_en	Transport block 2-layer split (soft modulation mode only): • 0 = Split Mode Disabled • 1 = Split Mode Enabled
1-0	sh_mod_sel	Modulation Mode Select: • 0 = Hard Modulation (uncompressed) • 1 = Hard Modulation (compressed) • 2 = Soft Modulation • 3 = n/a

4.6.3.3 CINIT_P2_CFG (Word 2)

Figure 4-69. Cinit P2 Value Config (Word 2)

31	30 0
Reserved	CINIT_P2

Table 4-74. Cinit P2 Value Config Details (Word 2)

Bit	Name	Description
31	Reserved	Value - NA
30-0	cinit_p2	C_{init} value for scrambling P2 sequence generator

4.6.3.4 RMUX_CQI_LN_CFG (Word 3) (Only used in soft modulation mode)

Figure 4-70. Rmux and CQI Length Config (Word 3)

31	30 16	15 6	5 0
Reserved	CQI_LN	Reserved	RMUX_LN

Table 4-75. Rmux and CQI Length Config Details

Bit	Name	Description
31	Reserved	Value - NA
30-16	cqi_ln	Value = 0 - 316 79 Number of CQI symbols to insert
15-6	Reserved	Value - NA
5-0	rmux_ln	Value = 0 - 34 R' $_{mux}$ length select: Selects 1 of 35 valid R' $_{mux}$ lengths depending on the setting of the split_mode_en field. The meaning of each value is given in Figure 4-59.

NOTE: Note that the case when a packet carries CQI data only (no payload information data) will be handled by the DSP and will not be sent for processing by the BCP MOD submodule.

Figure 4-71. R'$_{mux}$ Length Index Tables

Single layer

Index	N_{RB}	R'$_{mux}$
0	1	12
1	2	24
2	3	36
3	4	48
4	5	60
5	6	72
6	8	96
7	9	108
8	10	120
9	12	144
10	15	180
11	16	192
12	18	216
13	20	240
14	24	288
15	25	300
16	27	324
17	30	360
18	32	384
19	36	432
20	40	480
21	45	540
22	48	576
23	50	600
24	54	648
25	60	720
26	64	768
27	72	864
28	75	900
29	80	960
30	81	972
31	90	1080
32	96	1152
33	100	1200
34	108	1296

Dual layer

Index	N_{RB}	R'$_{mux}$
0	2	24
1	4	48
2	6	72
3	8	96
4	10	120
5	12	144
6	16	192
7	18	216
8	20	240
9	24	288
10	30	360
11	32	384
12	36	432
13	40	480
14	48	576
15	50	600
16	54	648
17	60	720
18	64	768
19	72	864
20	80	960
21	90	1080
22	96	1152
23	100	1200
24	108	1296
25	120	1440
26	128	1536
27	144	1728
28	150	1800
29	160	1920
30	162	1944
31	180	2160
32	192	2304
33	200	2400
34	216	2592

4.6.3.5 RI_ACK_LN_CFG (Word 4) (Only used in soft modulation mode)

Figure 4-72. RI and ACK Length Config (Word 4)

31 30	29 16	15 14	13 0
Reserved	ACK_LN	Reserved	RI_LN

Table 4-76. RI and ACK Length Config Details (Word 4)

Bit	Name	Description
31-30	Reserved	Value - NA
29-16	ack_ln	Value = 0 - 10560 Number of ACK symbols to overwrite
15-14	Reserved	Value - NA
13-0	ri_ln	Value = 0 - 10560 Number of RI symbols to insert

4.6.4 Error Bit Definitions

The MOD error bits are defined in Table 4-77. These are the errors bits that will be recorded in the MOD data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-77. MOD Error Bit Definitions

BIT	NAME
6	Reserved
5	Reserved
4	Reserved
3	Reserved
2	Reserved
1	Packet size error. Detected one of the following: • Not enough input data. • For Soft Modulation mode only, there is too much or too little total data to process (this includes input data as well as generated data e.g., CQI and RI). For example, there is an error if total data != cmux_sz * rmux_ln where cmux_sz = 9, 10 11 or 12.
0	Modulation mode error. Detected one of the following: • Compressed Hard Modulation mode selected AND 256 QAM selected • Soft Modulation mode selected AND WiMAX standard selected • Soft Modulation mode selected AND BPSK modulation selected • Soft Modulation mode selected AND rmux_ln > 34. • Un-available modulation type programmed i.e. 5, 6 or 7 • Unsupported Radio Standard

4.7 SSL

The Soft Slicer (SSL) submodule in the BCP performs soft demodulation as well as several other tasks. It receives packed symbol data in the form of modulation levels (i.e., I/Q for QAM modulation and PAM modulation levels for HSUPA) from the DSP. It outputs LLR soft bits (i.e., 1, 2, 4, 6, or 8 soft bits per symbol) that have been scaled by a noise scaling factors provided by the user. The noise scaling factors are sent as part of the packet header configuration fields for all modes except WiMAX, where the factors are included in the input data itself. In some use-cases, the outputs of the SSL go to the RD submodule for rate de-matching.

For LTE, the SSL submodule is used as part of the LTE uplink receiver. The SSL submodule performs partial de-interleaving, soft slicing (demodulation), descrambling, rank indication (RI) puncturing, and ACK/NACK overwrite operations. The SSL submodule's symbol input buffer is used as a ping-pong buffer. The partial de-interleaving operation is carried out by the way soft symbols are read from the input buffer. The symbols are read out of the input buffer and passed to the soft slicer logic to be demodulated. The soft slicer logic generates groups of Qm soft bits (one byte per soft bit). The soft bits are descrambled and then passed through the uplink control information (UCI) handler where RI soft bits are punctured and ACK/NACK soft bits are overwritten with all 0s.

The SSL submodule also supports WiMAX and HSUPA by only using the soft slicer function. In these modes, the de-interleaver, descrambler, and uplink control information handler are all bypassed and the input buffers are used as a FIFO.

The following modulation types are supported:

- BPSK (HS UPA FDD , Rel-99 UL FDD)
- 4-PAM (HS UPA FDD)
- QPSK (HS UPA TDD, LTE, WiMAX , Rel-99 UL TDD)
- 16-QAM (HS UPA TDD, LTE, WiMAX)
- 64-QAM (LTE, WiMAX)
- 256-QAM (LTE, WiMAX, not defined in standards)

For Rel -99 uplink, the SSL is configured for one physical channel (even if there are multiple Rel -99 physical channels) and by setting UVA all to the same value and the noise all to the same value. For Rel -99 FDD uplink, the modulation should be set to BPSK. For Rel -99 TDD uplink, the modulation should be set to QPSK . 8-PSK is not supported by the SSL submodule, so in that case demodulation must be done on the DSP before sending the data to the BCP DNT and RD submodules. The TTI length should be set to 10 ms. When used for Rel-99, the number of PAM symbols per segment should be computed as follows: wcdma_symb_seg = total number of PAM symbols in all Rel-99 physical channels / 5.

4.7.1 Data Format

4.7.1.1 LTE - Input Data

The natural ordering of the LTE receiver tasks based on the standards specification would be as follows:

Modulation de-mapping (Soft slicer) → Descrambling → Channel de-interleaving

However, since the memory size for the channel de-interleaver is rather large to be in the BCP, the de-interleaving operation is moved to be done first in the symbol domain (where "symbol" is equivalent to "subcarrier"). This allows the de-interleaver data to be stored outside the BCP in system memory. The decoding steps are then in the following order:

Channel de-interleaving on symbols (subcarriers) → Soft slicer → Descrambling

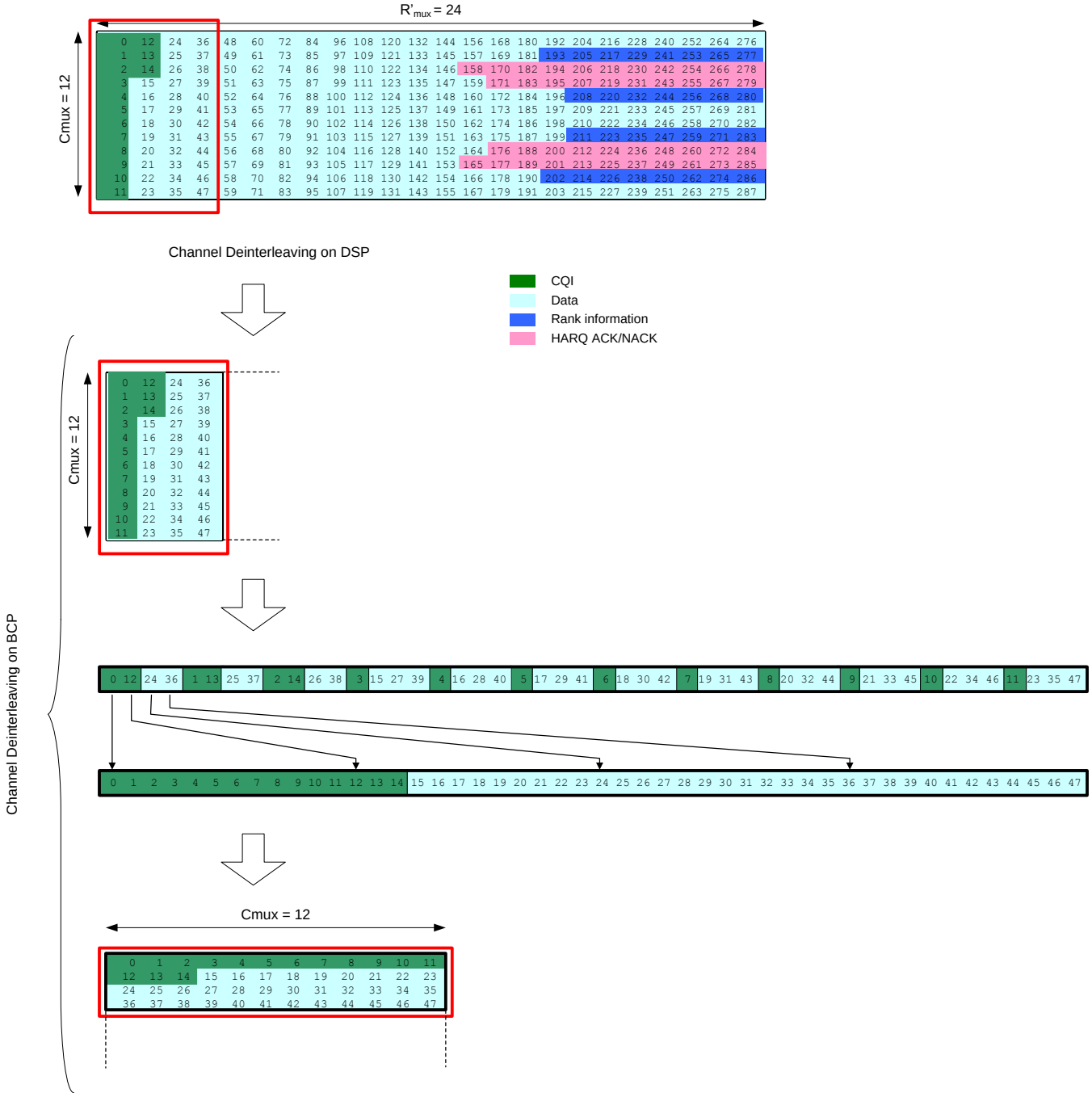
Each input data symbol is represented with 4 bytes (16-bit I and 16-bit Q). For efficiency, the de-interleaving function is split between the DSP and BCP. The DSP programs the EDMA to partially de-interleave the input data in groups of 4 data symbols (16 bytes) and then the BCP de-interleaves symbols within each segment of size 4×Cmux symbols (Cmux is the number of columns that was used in the original interleaver table). All those segments are sent to the BCP in a single packet. The decoding steps are then arranged in the following order:

1. Channel de-interleaving (stage 1) using EDMA
2. Channel de-interleaving (stage 2) in BCP
3. Soft slicing
4. Descrambling

The two-stage de-interleaving process is shown in Figure 4-60 where the number of columns in the original interleaver table Cmux = 12 and the number of rows R'mux = 24. The numbers in the matrix represent the indices of the de-interleaved data symbols. The red border represents the first block of 4 x Cmux symbols of data that would be sent in and processed.

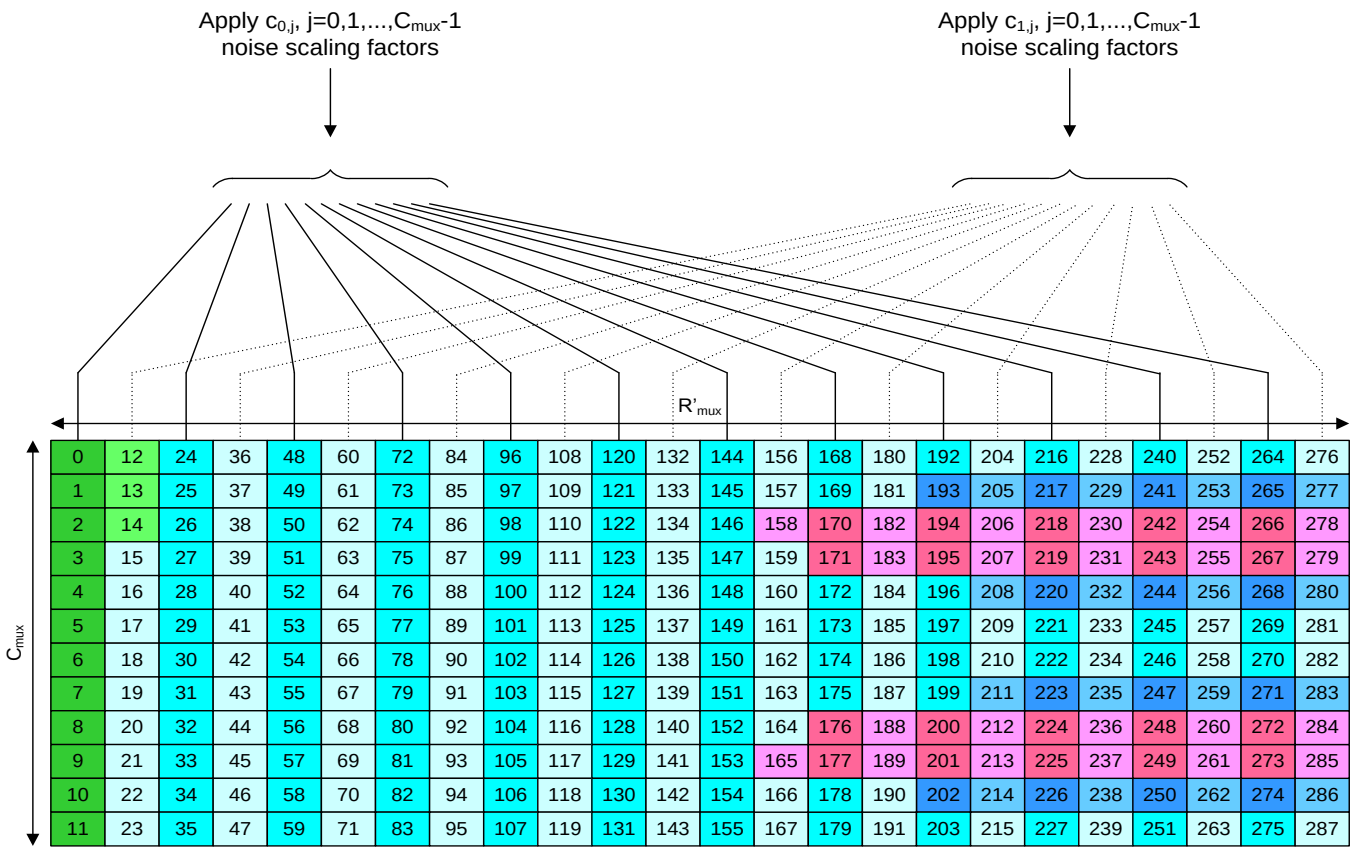
The top table represents the input data in the system memory outside the BCP. The DSP programs the EDMA to move the data in chunks of 4 symbols (16 bytes) as shown to accomplish the partial de-interleaving. The BCP then performs the rest of the de-interleaving within each block of 4 x Cmux symbols. The noise scaling factors c(i), where i=0 to Cmux-1, are applied to the corresponding row i in the top table.

Figure 4-73. Channel De-Interleaving Split Between DSP and BCP



The previous description applies to a regular scenario when a transport block is transmitted in one layer. In the case when a transport block is split into two layers, operations outlined in the previous figure apply the same way except that there are two sets of Cmux noise scaling factors. The other difference is that the maximum number of rows (R'mux) may be twice as large comparing to the regular scenario. Figure 4-61 illustrates the case when a transport block is split into two layers. The first layer (layer 0) is shown in darker colors. The layer mapping in the uplink is based on the principals of the layer mapping for the SU-MIMO case in the downlink.

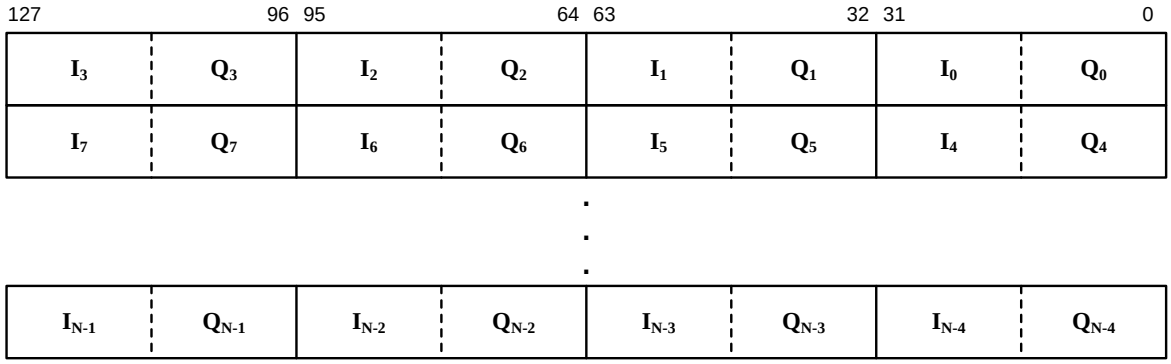
Figure 4-74. Transport Block Split into Two Layers



0	12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240	252	264	276
1	13	25	37	49	61	73	85	97	109	121	133	145	157	169	181	193	205	217	229	241	253	265	277
2	14	26	38	50	62	74	86	98	110	122	134	146	158	170	182	194	206	218	230	242	254	266	278
3	15	27	39	51	63	75	87	99	111	123	135	147	159	171	183	195	207	219	231	243	255	267	279
4	16	28	40	52	64	76	88	100	112	124	136	148	160	172	184	196	208	220	232	244	256	268	280
5	17	29	41	53	65	77	89	101	113	125	137	149	161	173	185	197	209	221	233	245	257	269	281
6	18	30	42	54	66	78	90	102	114	126	138	150	162	174	186	198	210	222	234	246	258	270	282
7	19	31	43	55	67	79	91	103	115	127	139	151	163	175	187	199	211	223	235	247	259	271	283
8	20	32	44	56	68	80	92	104	116	128	140	152	164	176	188	200	212	224	236	248	260	272	284
9	21	33	45	57	69	81	93	105	117	129	141	153	165	177	189	201	213	225	237	249	261	273	285
10	22	34	46	58	70	82	94	106	118	130	142	154	166	178	190	202	214	226	238	250	262	274	286
11	23	35	47	59	71	83	95	107	119	131	143	155	167	179	191	203	215	227	239	251	263	275	287

The bit format of the input data in either case is as follows.

Figure 4-75. SSL Input Bit Format for LTE

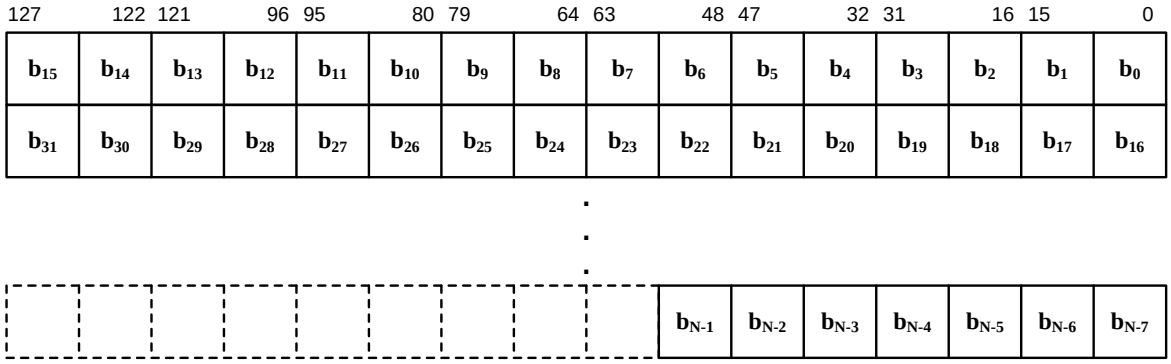


The order of the I and Q components may be reversed by using the “jack_bit” field in the packet header configuration.

4.7.1.2 LTE - Output Data

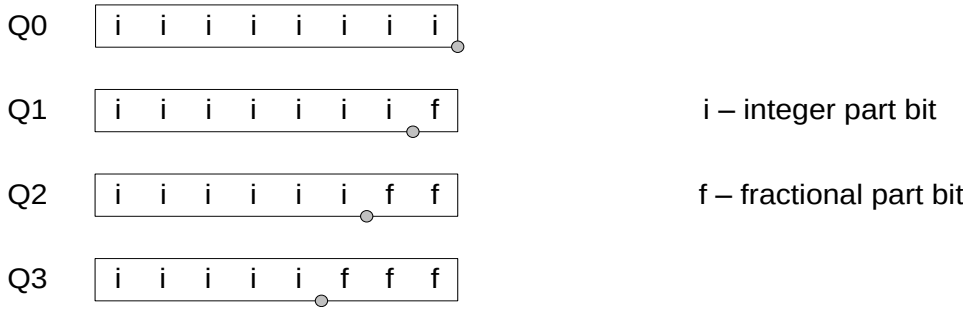
The output of the SSL submodule is 8-bit LLR soft bits. The bit format for LTE is shown in Figure 4-76.

Figure 4-76. SSL Output Data Bit Format for LTE



The precision of these soft bits is programmable on a per packet basis via the q_format field of the MOD_SEL_CFG word in the packet configuration header. The available Q formats are shown in Figure 4-66.

Figure 4-77. SSL Output Data Precision



4.7.1.3 HSPA - Input Data

For HSPA, the SSL submodule may be used to perform only the soft slicing while the de-interleaving and descrambling are disabled.

The input data bit formats for HSPA are shown in the following two figures. Figure 4-78 is for HSPA FDD and Figure 4-79 is for HSPA TDD.

Figure 4-78. SSL Input Bit Format for HSPA FDD PAM Modulation Levels (16-bit)

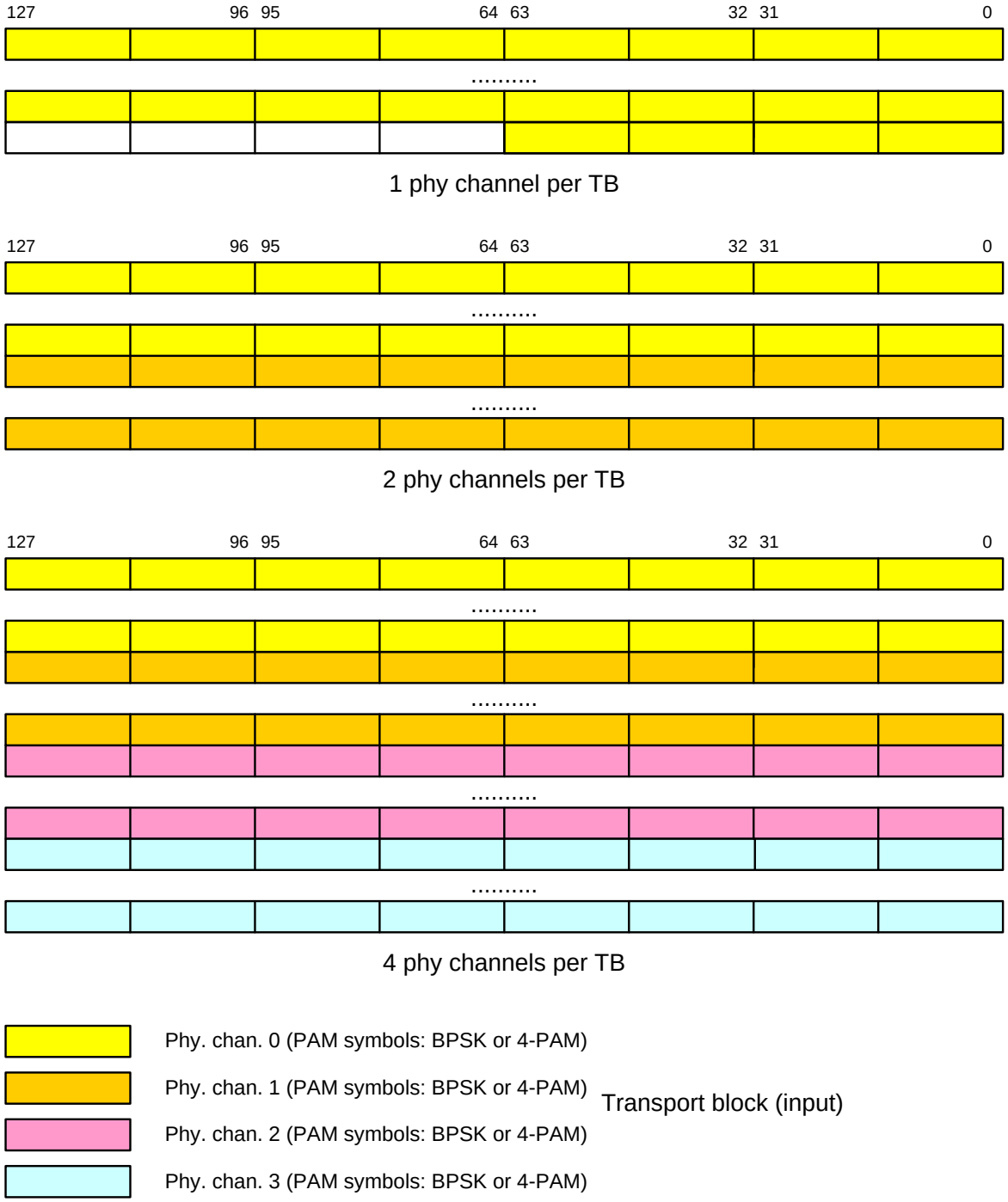
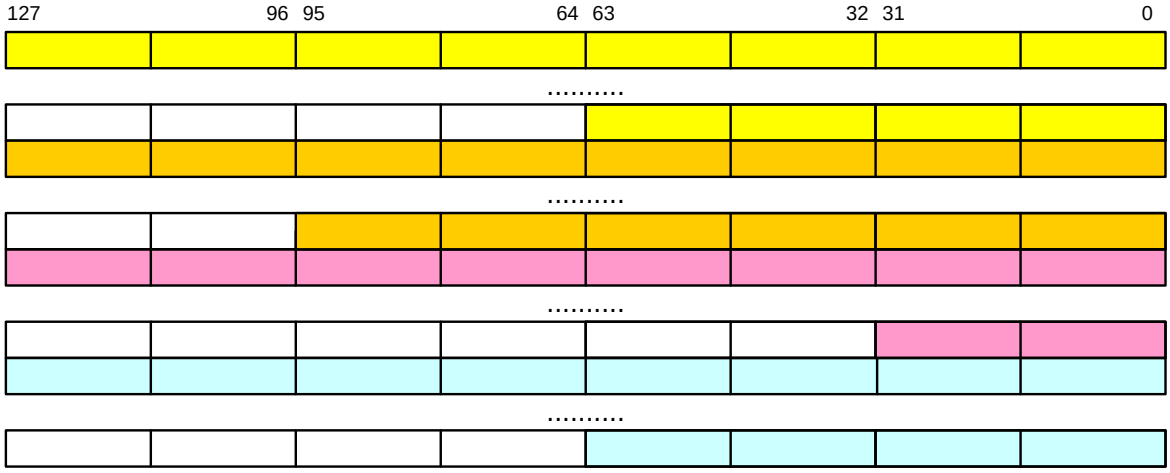


Figure 4-79. HSPA TDD QAM Modulation (16-bit I, 16-bit Q)



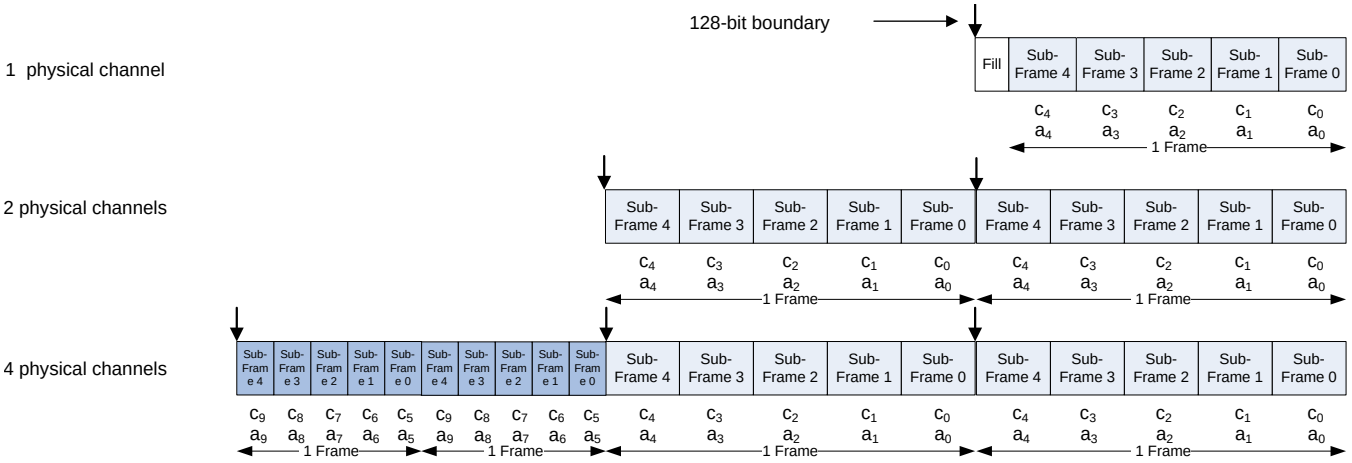
The order of the I and Q components is controlled by the “jack_bit” field in the packet header configuration.

In HSPA TDD, each time slot has its own noise scaling factor and QAM modulation mode.

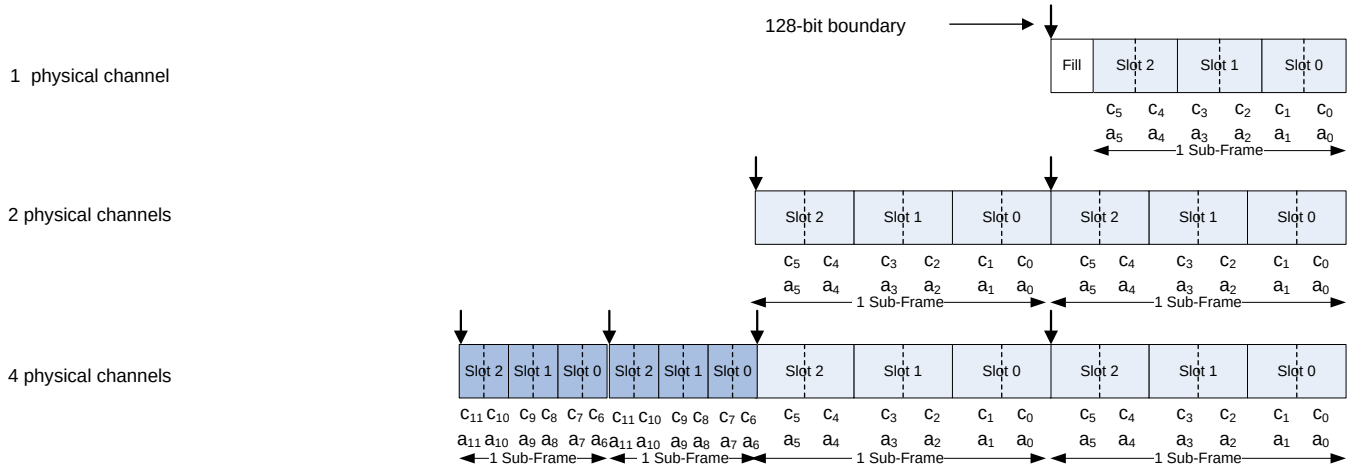
In HSPA FDD case, the noise scaling factors are each applied to a group of PAM symbols, as shown in Figure 4-80. The size of the group depends on the TTI length.

- 10ms TTI – One noise scaling factor per sub-frame
- 2ms TTI – One noise scaling factor per half slot

Figure 4-80. Noise Scaling Factor Application for HSPA FDD



a. One noise scaling factor and one unit value per sub-frame (10ms)



b. One noise scaling factor and one unit value per half slot (2ms)

4.7.1.4 HSPA - Output Data

The output of the SSL submodule is 8-bit LLR soft bits. The output data bit formats for HSPA are shown in the following two figures. Figure 4-81 is for HSPA FDD and Figure 4-74 is for HSPA TDD.

Figure 4-81. SSL Output Bit Format for HSPA FDD

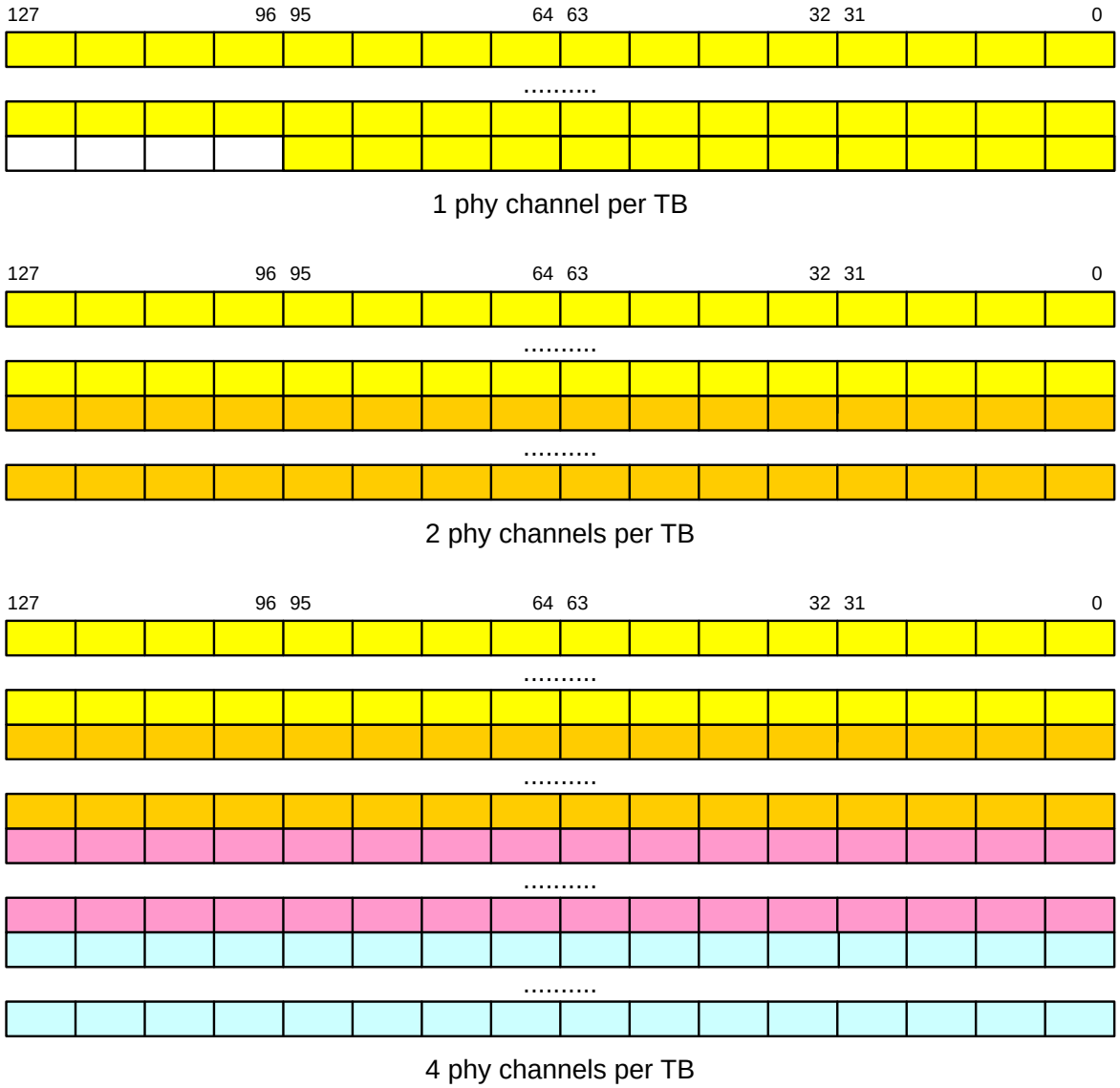
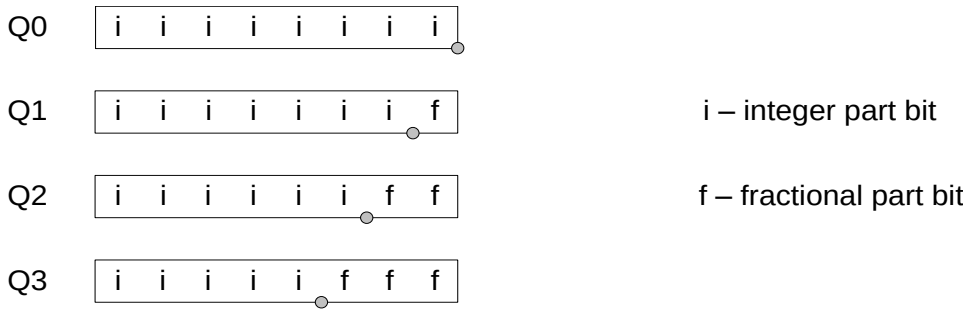


Figure 4-82. SSL Output Bit format for HSPA TDD

As shown in the HSPA TDD figure, the soft bits belonging to all time slots may be either 128-bit aligned (upper part of the figure) or packed (lower part of the figure).

The precision of the output soft bits is programmable on a per packet basis via the q_format field of the MOD_SEL_CFG word in the packet configuration header. The available Q formats are shown in Figure 4-83.

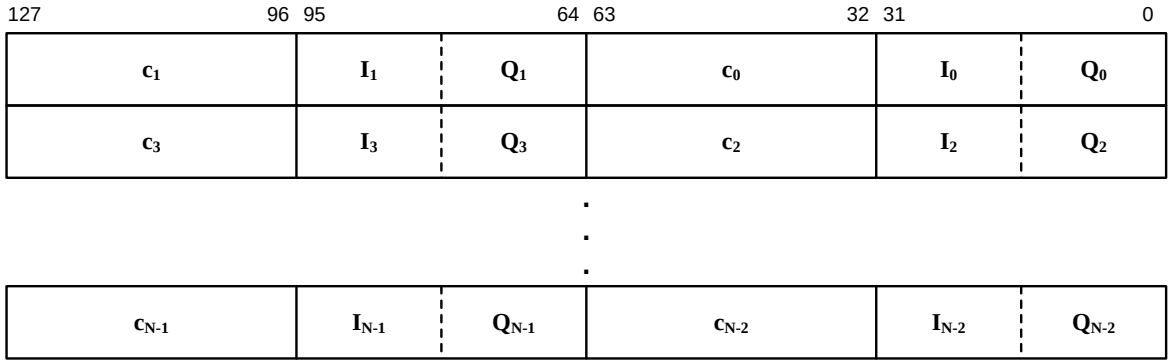
Figure 4-83. SSL Output Data Precision



4.7.1.5 WiMAX

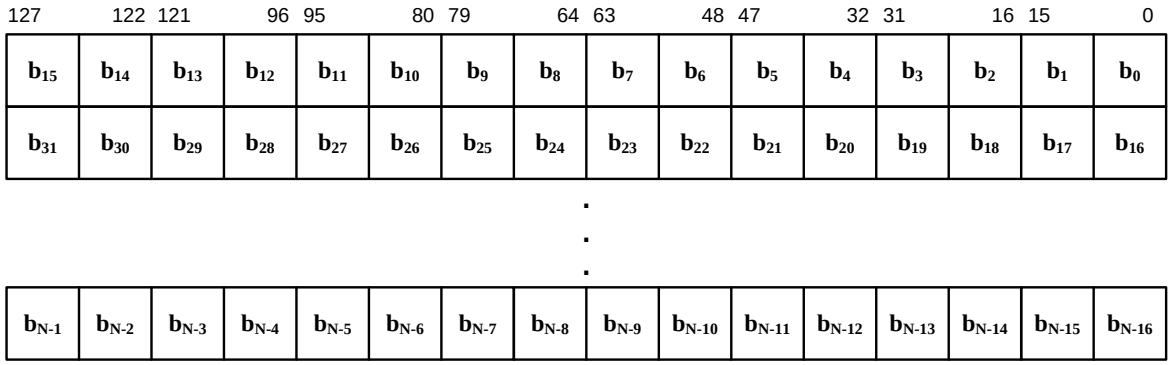
For WiMAX, the SSL submodule may be used to perform only the soft slicing while the de-interleaving and descrambling are disabled. Each QAM symbol is provided with its own noise scaling factor, as shown in Figure 4-84. The order of the I and Q components is controlled by the “jack_bit” field in the packet header configuration.

Figure 4-84. WiMAX QAM Modulation with Noise Scaling Factors per Symbol (16-bit I, 16-bit Q)



The output of the SSL submodule is 8-bit LLR soft bits. The output data bit format for WiMAX is shown in Figure 4-85.

Figure 4-85. SSL Output Bit Format for WiMAX



4.7.2 Memory Mapped Registers

The only memory mapped registers are the data logger registers and are described in the data logger section.

4.7.3 Packet Header Configuration Fields

The order of the packet configuration fields used for each standard is shown in the following figures.

Figure 4-86. Single Layer LTE Packet Header (28 words)

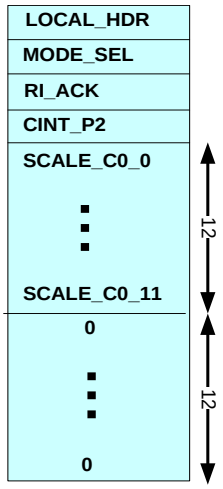


Figure 4-87. Split Mode LTE Packet Header (28 words)

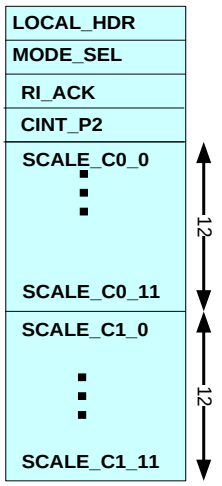


Figure 4-88. WCDMA (FDD) Packet Header (20 words)

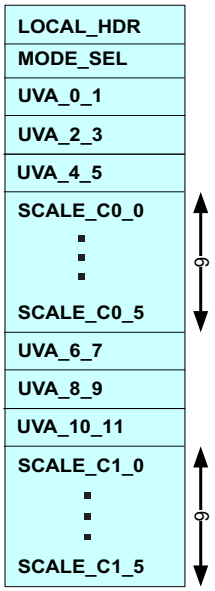


Figure 4-89. TD-SCDMA Packet Header (14 words)

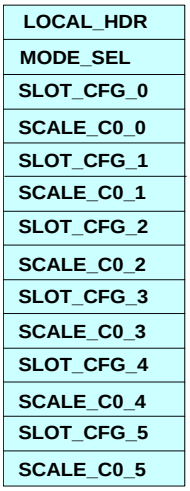


Figure 4-90. WiMax Packet Header (28 words)

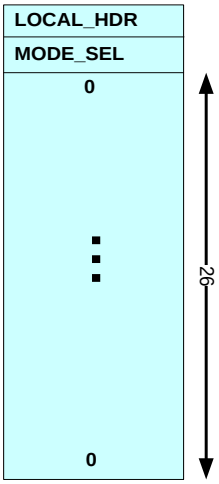


Table 4-78 provides a description of each configuration field.

4.7.3.1 LOCAL_CFG_HDR

Figure 4-91. Local Packet Configuration Header Register

31 16	15 12	11 8	7 6	0
See Table 4-78	Reserved	MOD_ID	Reserved	LOCAL_HDR_LEN

Table 4-78. Local Packet Configuration Header Register Details

Bit	Name	Description
LTE & WIMAX		
31-16	UVA	LTE & WIMAX: Soft Slicer Unit Value (equivalent of 1): • 2-PAM: 0-32767 (QPSK & BPSK) • 4-PAM: 0-10922 (16 QAM and 4-PAM) • 6-PAM: 0-4681 (64 QAM) • 8-PAM: 0-2184 (256 QAM)
WCDMA (FDD)		
31-30	Reserved	Reserved
29-16	WCDMA_SYMB_SEG	WCDMA (FDD): WCDMA Number of Symbols per Segment
WCDMA (TDD)		
31-16	Reserved	WCDMA (TDD): Reserved
All		
15-12	Reserved	Reserved
11-8	Mod_id	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	Local_hdr_len	Local header length in 32-bit words, excluding this word (Word 0).

4.7.3.2 MODE_SEL_CFG

Figure 4-92. Mode Select Config Register

31 26	25 23	22 20	19 18	12	11 10
RMUX_LN _INDEX	WCDMA_NUM_PHY_CH	WCDMA_NUM_SLOTS	TTI_2MS_ 10MS_SEL	B_MATRIX_SEL	Q_FORMAT

9 8	7 6	4	3 2	1	0
CMUX_LN	TDD_OUTPUT _PACK	MOD_TYPE _SEL	LTE_DESCRAMBLER _EN	JACK_BIT	SPLIT_MODE _EN	FDD_TDD _SELECT

Table 4-79. Mode Select Config Register Details

Bit	Name	Description
31-26	rmux_ln_index	Value - 0 - 34 R_{mux} length Index (LTE only): Index into rmux_length tables. There are two tables one for Split Mode and one for Normal Mode. See Figure 4-63.
25-23	wcdma_num_phy_ch	Value = 1,2,4 Number of physical channels for WCDMA (FDD)
22-20	wcdma_num_slots	Number of WCDMA (TD-SCDMA) slots – maximum of 6
19	tti_2ms_10ms_sel	WCDMA(FDD) only • 0 = 2ms TTI • 1 = 10ms TTI
11-10	q_format	LLR Q Format: • 0 = Q0 • 1 = Q1 • 2 = Q2 • 3 = Q3
9-8	cmux_ln	C_{mux} length select (LTE only): • 0 = 9 Columns (Extended w/ SRS) • 1 = 10 Columns (Extended w/o SRS) • 2 = 11 Columns (Normal w/ SRS) • 3 = 12 Columns (Normal w/o SRS)
7	tdd_output_pack	WCDMA TDD output from SSL slot packing mode (Only used in WCDMA (Rel-99/HSPA) TDD mode) • 0: slots aligned on 128 bit boundaries • 1: slots not aligned on 128 bit boundaries i.e. packed
6-4	mod_type_sel	Modulation Type Select: • 0 = BPSK - WCDMA(FDD) only • 1 = QPSK - LTE, WiMax, WCDMA (TDD) only • 2 = 16 QAM - LTE, WiMax. WCDMA (TDD) only • 3 = 64 QAM - LTE, WiMax, WCDMA (TDD) only • 4 = 256 QAM - LTE, WiMax only • 5 = 4 PAM - WCDMA(FDD) only • 6 -7 = n/a
3	lte_descrambler_en	LTE Descrambling Enable: • 0 = Descrambling Bypassed • 1 = Descrambling Enabled This bit is only provided as a debug mechanism for LTE applications. It will always be set to 1 for normal LTE operation. It is ignored for WiMAX and WCDMA/TD-SCDMA.
2	jack_bit	IQ Output Reverse: • 0 = No IQ reverse • 1 = Reverse IQ This bit is used to correct for input IQ data that is formatted in reverse for LTE, WiMax, and WCDMA (TD-SCDMA). When set to a 1, it does not affect the format of the WiMAX scaling factors or WCDMA (FDD) PAM modulation levels.
1	split_mode_en	Transport block 2-layer split (LTE only): • 0 = Split Mode Disabled • 1 = Split Mode Enabled
0	fdd_tdd_select	• 0 = FDD selected • 1 = TDD selected When the radio_standard is WCDMA R99 or WCDMA HSPA this bit indicates if WCDMA(FDD) is selected or WCDMA TDSCDMA is selected.

Figure 4-93. R'_{mux} Length Index Tables

Single layer

Index	N_{RB}	R'_{mux}
0	1	12
1	2	24
2	3	36
3	4	48
4	5	60
5	6	72
6	8	96
7	9	108
8	10	120
9	12	144
10	15	180
11	16	192
12	18	216
13	20	240
14	24	288
15	25	300
16	27	324
17	30	360
18	32	384
19	36	432
20	40	480
21	45	540
22	48	576
23	50	600
24	54	648
25	60	720
26	64	768
27	72	864
28	75	900
29	80	960
30	81	972
31	90	1080
32	96	1152
33	100	1200
34	108	1296

Dual layer

Index	N_{RB}	R'_{mux}
0	2	24
1	4	48
2	6	72
3	8	96
4	10	120
5	12	144
6	16	192
7	18	216
8	20	240
9	24	288
10	30	360
11	32	384
12	36	432
13	40	480
14	48	576
15	50	600
16	54	648
17	60	720
18	64	768
19	72	864
20	80	960
21	90	1080
22	96	1152
23	100	1200
24	108	1296
25	120	1440
26	128	1536
27	144	1728
28	150	1800
29	160	1920
30	162	1944
31	180	2160
32	192	2304
33	200	2400
34	216	2592

The following code may be used to calculate the “b_matrix_sel” input packet header field.

Example 4-1. Example code to calculate B-matrix index parameter b_matrix_sel

```
// Inputs
UInt32 Rprime;      // Index of prbList table corresponding to current number of
                    // RBs allocated

UInt8  mod;         // Modulation type (Qm)
UInt8  numLayers;   // Number of layers (1 or 2)

// Output
Uint8 bTableIndex;  // B-matrix index for MOD and SSL blocks

// Intermediate variable
UInt32 Rprime1;     // Number of subcarriers allocated

// PRB table for single layer (each element is multiplied by a factor of 2 to
// get table for dual-layer when needed)

UInt8 prbList[35] =
{
    1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16,
    18, 20, 24, 25, 27, 30, 32, 36, 40, 45,
    48, 50, 54, 60, 64, 72, 75, 80, 81, 90,
    96, 100, 108
};

// Mapping table used to find B-matrix index
Uint32 rowTbl[72] = {

      24,      48,      72,      96,     120,     144,     192,     216,
     240,     288,     360,     384,     432,     480,     576,     600,
     648,     720,     768,     864,     960,    1080,    1152,    1200,
    1296,    1440,    1536,    1728,    1800,    1920,    1944,    2160,
    2304,    2400,    2592,    2880,    3072,    3240,    3456,    3600,
    3840,    3888,    4320,    4608,    4800,    5184,    5400,    5760,
    5832,    6144,    6480,    6912,    7200,    7680,    7776,    8640,
    9216,    9600,   10368,   10800,   11520,   11664,   12288,   12960,
   13824,   14400,   15360,   15552,   17280,   18432,   19200,   20736};

// Utility function to calculate modulation engine's B table index.
// (Finds value in rowTbl[] and returns corresponding index)

UInt8 findRowIndex (UInt32 input)
{
 UInt8 i;

 for (i = 0; i < 72; i++)
 {
   if (rowTbl[i] == input)

    return i;
 }
  return 0;
}

// Calculate number of subcarriers allocated
Rprime1 = 12 * prbList[Rprime] * numLayers;

// Find B-matrix index
bTableIndex = findRowIndex(Rprime1 * mod);
```

4.7.3.3 RI_ACK_LN_CFG

Figure 4-94. RI and ACK Length Config Register

31 30	29 16	15 14	13 0
Reserved	ACK_LN	Reserved	RI_LN

Table 4-80. RI and ACK Length Config Register Details

Bit	Name	Description
31-30	Reserved	Value = NA
29-16	ack_ln	Number of ACK LLRs of Q_M soft bits to overwrite: LTE only • 0 – 5184 Normal mode • 0- 10368 Split mode
15-14	Reserved	Reserved
13-0	ri_ln	Number of RI LLRs of Q_M soft bits to puncture: LTE only • 0 – 5184 Normal mode • 0- 10368 Split mode

4.7.3.4 WCDMA_SLOT_CFG_0

Figure 4-95. WCDMA (TD-SCDMA) Slot 0 Configuration Register

31 16	15 12	11 0
UVA_0	Reserved	WCDMA_SIZE_SLOT_0

Table 4-81. WCDMA (TD-SCDMA) Slot 0 Configuration Register Details

Bit	Name	Description
31-16	uva_0	WCDMA (TD-SCDMA): Soft Slicer Unit Value (equivalent of 1) for slot 0: • 2-PAM: 0-16383 (QPSK) • 4-PAM: 0-10922 (16 QAM) • 6-PAM: 0-4681 (64 QAM)
15-12	Reserved	Value = NA
11-0	wcdma_size_slot_0	WCDMA (TD-SCDMA) number of symbols for slot 0 Max size 3064

4.7.3.5 WCDMA_SLOT_CFG_1

Figure 4-96. WCDMA (TD-SCDMA) Slot 1 Configuration Register

31 16	15 12	11 0
UVA_1	Reserved	WCDMA_SIZE_SLOT_1

Table 4-82. WCDMA (TD-SCDMA) Slot 1 Configuration Register Details

Bit	Name	Description
31-16	uva_1	WCDMA (TD-SCDMA): Soft Slicer Unit Value (equivalent of 1) for slot 1: • 2-PAM: 0-16383 (QPSK) • 4-PAM: 0-10922 (16 QAM) • 6-PAM: 0-4681 (64 QAM)
15-12	Reserved	Value = NA
11-0	wcdma_size_slot_1	WCDMA (TD-SCDMA) number of symbols for slot 1 Max size 3064

4.7.3.6 WCDMA_SLOT_CFG_2

Figure 4-97. WCDMA (TD-SCDMA) Slot 2 Configuration Register

31 16	15 12	11 0
UVA_2	Reserved	WCDMA_SIZE_SLOT_2

Table 4-83. WCDMA (TD-SCDMA) Slot 2 Configuration Register Details

Bit	Name	Description
31-16	uva_2	WCDMA (TD-SCDMA): Soft Slicer Unit Value (equivalent of 1) for slot 2: • 2-PAM: 0-16383 (QPSK) • 4-PAM: 0-10922 (16 QAM) • 6-PAM: 0-4681 (64 QAM)
15-12	Reserved	Value = NA
11-0	wcdma_size_slot_2	WCDMA (TD-SCDMA) number of symbols for slot 2 Max size 3064

4.7.3.7 WCDMA_SLOT_CFG_3

Figure 4-98. WCDMA (TD-SCDMA) Slot 3 Configuration Register

31 16	15 12	11 0
UVA_3	Reserved	WCDMA_SIZE_SLOT_3

Table 4-84. WCDMA (TD-SCDMA) Slot 3 Configuration Register Details

Bit	Name	Description
31-16	uva_3	WCDMA (TD-SCDMA): Soft Slicer Unit Value (equivalent of 1) for slot 3: • 2-PAM: 0-16383 (QPSK) • 4-PAM: 0-10922 (16 QAM) • 6-PAM: 0-4681 (64 QAM)
15-12	Reserved	Value = NA
11-0	wcdma_size_slot_3	WCDMA (TD-SCDMA) number of symbols for slot 3 Max size 3064

4.7.3.8 WCDMA_SLOT_CFG_4

Figure 4-99. WCDMA (TD-SCDMA) Slot 4 Configuration Register

31 16	15 12	11 0
UVA_4	Reserved	WCDMA_SIZE_SLOT_4

Table 4-85. WCDMA (TD-SCDMA) Slot 4 Configuration Register Details

Bit	Name	Description
31-16	uva_4	WCDMA (TD-SCDMA): Soft Slicer Unit Value (equivalent of 1) for slot 4: • 2-PAM: 0-16383 (QPSK) • 4-PAM: 0-10922 (16 QAM) • 6-PAM: 0-4681 (64 QAM)
15-12	Reserved	Value = NA
11-0	wcdma_size_slot_4	WCDMA (TD-SCDMA) number of symbols for slot 4 Max size 3064

4.7.3.9 WCDMA_SLOT_CFG_5

Figure 4-100. WCDMA (TD-SCDMA) Slot 5 Configuration Register

31 16	15 12	11 0
UVA_5	Reserved	WCDMA_SIZE_SLOT_5

Table 4-86. WCDMA (TD-SCDMA) Slot 5 Configuration Register Details

Bit	Name	Description
31-16	uva_5	WCDMA (TD-SCDMA): Soft Slicer Unit Value (equivalent of 1) for slot 5: • 2-PAM: 0-16383 (QPSK) • 4-PAM: 0-10922 (16 QAM) • 6-PAM: 0-4681 (64 QAM)
15-12	Reserved	Value = NA
11-0	wcdma_size_slot_5	WCDMA (TD-SCDMA) number of symbols for slot 5 Max size 3064

4.7.3.10 WCDMA_FDD_UVA_0_1

Figure 4-101. WCDMA (FDD) UVA 0&1 Configuration Register

31 16	15 0
UVA_0	UVA_1

Table 4-87. WCDMA (FDD) UVA 0&1 Configuration Register Details

Bit	Name	Description
31-16	uva_0	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 0: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)
15-0	uva_1	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 1: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)

4.7.3.11 WCDMA_FDD_UVA_2_3

Figure 4-102. WCDMA (FDD) UVA 2&3 Configuration Register

31 16	15 0
UVA_2	UVA_3

Table 4-88. WCDMA (FDD) UVA 2&3 Configuration Register Details

Bit	Name	Description
31-16	uva_2	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 2: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)
15-0	uva_3	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 3: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)

4.7.3.12 WCDMA_FDD_UVA_4_5

Figure 4-103. WCDMA (FDD) UVA 4&5 Configuration Register

31 16	15 0
UVA_4	UVA_5

Table 4-89. WCDMA (FDD) UVA 4&5 Configuration Register Details

Bit	Name	Description
31-16	uva_4	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 4: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)
15-0	uva_5	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 5: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)

4.7.3.13 WCDMA_FDD_UVA_6_7

Figure 4-104. WCDMA (FDD) UVA 6&7 Configuration Register

31 16	15 0
UVA_6	UVA_7

Table 4-90. WCDMA (FDD) UVA 6&7 Configuration Register Details

Bit	Name	Description
31-16	uva_6	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 6: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)
15-0	uva_7	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 7: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)

4.7.3.14 WCDMA_FDD_UVA_8_9

Figure 4-105. WCDMA (FDD) UVA 8&9 Configuration Register

31 16	15 0
UVA_8	UVA_9

Table 4-91. WCDMA (FDD) UVA 8&9 Configuration Register Details

Bit	Name	Description
31-16	uva_8	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 8: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)
15-0	uva_9	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 9: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)

4.7.3.15 WCDMA_FDD_UVA_10_11

Figure 4-106. WCDMA (FDD) UVA 8&9 Configuration Register

31 16	15 0
UVA_10	UVA_11

Table 4-92. WCDMA (FDD) UVA 10&11 Configuration Register Details

Bit	Name	Description
31-16	uva_10	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 10: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)
15-0	uva_11	WCDMA (FDD): Soft Slicer Unit Value (equivalent of 1) for slot 11: • 2-PAM: 0-16383 (BPSK) • 4-PAM: 0-10922 (4 PAM)

4.7.3.16 SCALE_C0_0_CFG

Figure 4-107. Column0_0 Noise Scaling Factor Config Register

31 0
SCALE_C0_0

Table 4-93. Column0_0 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_0	Noise scaling factor (IEEE 754 Format) for: • Column0 in LTE single layer mode • Column 0,0 in LTE split mode • Slot0 in WCDMA (TD-SCDMA)) • Segment 0 in WCDMA (FDD) (1) Not used for WiMax

4.7.3.17 SCALE_C0_1_CFG

Figure 4-108. Column0_1 Noise Scaling Factor Config Register

31 0
SCALE_C0_1

Table 4-94. Column0_1 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_1	Noise scaling factor (IEEE 754 Format) for: • Column1 in LTE single layer mode • Column 0,1 in LTE split mode • Slot1 in WCDMA (TD-SCDMA)) • Segment 1 in WCDMA (FDD) (1) Not used for WiMax

4.7.3.18 SCALE_C0_2_CFG

Figure 4-109. Column0_2 Noise Scaling Factor Config Register

Table 4-95. Column0_2 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_2	Noise scaling factor (IEEE 754 Format) for: • Column2 in LTE single layer mode • Column 0,2 in LTE split mode • Slot2 in WCDMA (TD-SCDMA)) • Segment 2 in WCDMA (FDD) (1) Not used for WiMax

4.7.3.19 SCALE_C0_3_CFG

Figure 4-110. Column0_3 Noise Scaling Factor Config Register

Table 4-96. Column0_3 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_3	Noise scaling factor (IEEE 754 Format) for: • Column3 in LTE single layer mode • Column 0,3 in LTE split mode • Slot3 in WCDMA (TD-SCDMA) • Segment 3 in WCDMA (FDD) (1) Not used for WiMax

4.7.3.20 SCALE_C0_4_CFG

Figure 4-111. Column0_4 Noise Scaling Factor Config Register

Table 4-97. Column0_4 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_4	Noise scaling factor (IEEE 754 Format) for: • Column4 in LTE single layer mode • Column 0,4 in LTE split mode • Slot4 in WCDMA (TD-SCDMA) • Segment 4 in WCDMA (FDD) (1) Not used for WiMax

4.7.3.21 SCALE_C0_5_CFG

Figure 4-112. Column0_5 Noise Scaling Factor Config Register

31	0
SCALE_C0_5	

Table 4-98. Column0_5 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_5	Noise scaling factor (IEEE 754 Format) for: • Column5 in LTE single layer mode • Column 0,5 in LTE split mode • Slot5 in WCDMA (TD-SCDMA) • Segment 5 in WCDMA (FDD) (1) Not used for WiMax

4.7.3.22 SCALE_C0_6_CFG

Figure 4-113. Column0_6 Noise Scaling Factor Config Register

31	0
SCALE_C0_6	

Table 4-99. Column0_6 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_6	Noise scaling factor (IEEE 754 Format) for: • Column6 in LTE single layer mode • Column 0,6 in LTE split mode • Segment 6 in WCDMA (FDD) (1) Not used for WiMax or WCDMA/TD-SCDMA

4.7.3.23 SCALE_C0_7_CFG

Figure 4-114. Column0_7 Noise Scaling Factor Config Register

31	0
SCALE_C0_7	

Table 4-100. Column0_7 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_7	Noise scaling factor (IEEE 754 Format) for: • Column7 in LTE single layer mode • Column 0,7 in LTE split mode • Segment 7 in WCDMA (FDD) (1) Not used for WiMax or WCDMA/TD-SCDMA

4.7.3.24 SCALE_C0_8_CFG

Figure 4-115. Column0_8 Noise Scaling Factor Config Register

31	0
SCALE_C0_8	

Table 4-101. Column0_8 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_8	Noise scaling factor (IEEE 754 Format) for: • Column8 in LTE single layer mode • Column 0,8 in LTE split mode • Segment 8 in WCDMA (FDD) (1) Not used for WiMax or WCDMA/TD-SCDMA

4.7.3.25 SCALE_C0_9_CFG

Figure 4-116. Column0_9 Noise Scaling Factor Config Register

31	0
SCALE_C0_9	

Table 4-102. Column0_9 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_9	Noise scaling factor (IEEE 754 Format) for: • Column9 in LTE single layer mode if cmux_ln >9 • Column 0,9 in LTE split mode if cmux_ln >9 • Segment 9 in WCDMA (FDD) (1) Not used for WiMax or WCDMA/TD-SCDMA or LTE when cmux_ln=9

4.7.3.26 SCALE_C0_10_CFG

Figure 4-117. Column0_10 Noise Scaling Factor Config Register

31	0
SCALE_C0_10	

Table 4-103. Column0_10 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_10	Noise scaling factor (IEEE 754 Format) for: • Column10 in LTE single layer mode if cmux_ln >10 • Column 0,10 in LTE split mode if cmux_ln >10 • Segment 10 in WCDMA (FDD) (1) Not used for WiMax or WCDMA/TD-SCDMA or LTE when cmux_ln <11

4.7.3.27 SCALE_C0_11_CFG

Figure 4-118. Column0_11 Noise Scaling Factor Config Register

31	0
SCALE_C0_11	

Table 4-104. Column0_11 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c0_11	Noise scaling factor (IEEE 754 Format) for: • Column11 in LTE single layer mode if cmux_ln =12 • Column 0,11 in LTE split mode if cmux_ln =12 • Segment 11 in WCDMA (FDD) (1) Not used for WiMax or WCDMA/TD-SCDMA or LTE when cmux_ln <12

4.7.3.28 SCALE_C1_0_CFG

Figure 4-119. Column1_0 Noise Scaling Factor Config Register

31	0
SCALE_C1_0	

Table 4-105. Column1_0 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_0	Noise scaling factor (IEEE 754 Format) for: Column 1,0 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.29 SCALE_C1_1_CFG

Figure 4-120. Column1_1 Noise Scaling Factor Config Register

31	0
SCALE_C1_1	

Table 4-106. Column1_1 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_1	Noise scaling factor (IEEE 754 Format) for: Column 1,1 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.30 SCALE_C1_2_CFG

Figure 4-121. Column1_2 Noise Scaling Factor Config Register

31	0
SCALE_C1_2	

Table 4-107. Column1_2 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_2	Noise scaling factor (IEEE 754 Format) for: Column 1,2 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.31 SCALE_C1_3_CFG

Figure 4-122. Column1_3 Noise Scaling Factor Config Register

Table 4-108. Column1_3 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_3	Noise scaling factor (IEEE 754 Format) for: Column 1,3 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.32 SCALE_C1_4_CFG

Figure 4-123. Column1_4 Noise Scaling Factor Config Register

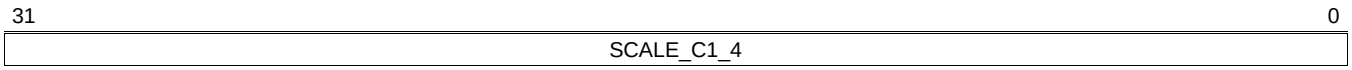


Table 4-109. Column1_4 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_4	Noise scaling factor (IEEE 754 Format) for: Column 1,4 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.33 SCALE_C1_5_CFG

Figure 4-124. Column1_5 Noise Scaling Factor Config Register

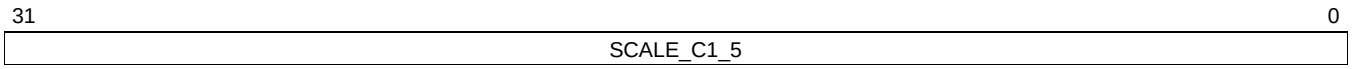


Table 4-110. Column1_5 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_5	Noise scaling factor (IEEE 754 Format) for: Column 1,5 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.34 SCALE_C1_6_CFG

Figure 4-125. Column1_6 Noise Scaling Factor Config Register

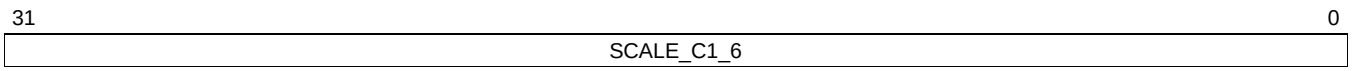


Table 4-111. Column1_6 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_6	Noise scaling factor (IEEE 754 Format) for: Column 1,6 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.35 SCALE_C1_7_CFG

Figure 4-126. Column1_7 Noise Scaling Factor Config Register

Table 4-112. Column1_7 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_7	Noise scaling factor (IEEE 754 Format) for: Column 1,7 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.36 SCALE_C1_8_CFG

Figure 4-127. Column1_8 Noise Scaling Factor Config Register

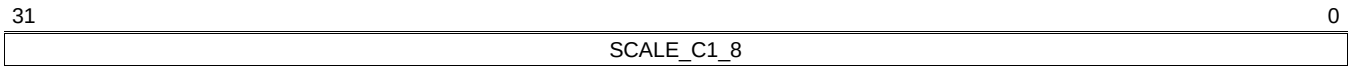


Table 4-113. Column1_8 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_8	Noise scaling factor (IEEE 754 Format) for: Column 1,8 in LTE split mode Not used for WiMax, WCDMA or single layer LTE

4.7.3.37 SCALE_C1_9_CFG

Figure 4-128. Column1_9 Noise Scaling Factor Config Register

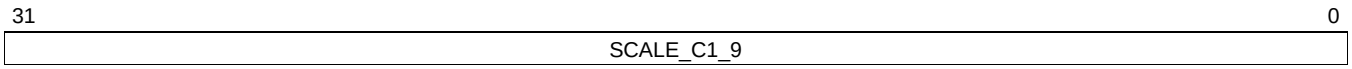


Table 4-114. Column1_9 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_9	Noise scaling factor (IEEE 754 Format) for: Column 1,9 in LTE split mode if cmux_ln >9 Not used for WiMax, WCDMA, single layer LTE or split mode LTE if cmux_ln=9

4.7.3.38 SCALE_C1_10_CFG

Figure 4-129. Column1_10 Noise Scaling Factor Config Register

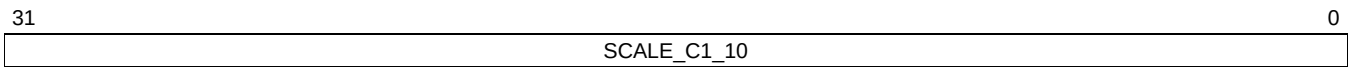


Table 4-115. Column1_10 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_10	Noise scaling factor (IEEE 754 Format) for: Column 1,10 in LTE split mode if cmux_ln>10 Not used for WiMax, WCDMA, single layer LTE or split mode LTE if cmux_ln<11

4.7.3.39 SCALE_C1_11_CFG

Figure 4-130. Column1_11 Noise Scaling Factor Config Register

31	0
SCALE_C1_11	

Table 4-116. Column1_11 Noise Scaling Factor Config Register Details

Bit	Name	Description
31-0	scale_c1_11	Noise scaling factor (IEEE 754 Format) for: Column 1,11 in LTE split mode if cmux_ln=12 Not used for WiMax, WCDMA, single layer LTE or split mode LTE if cmux_ln<12

NOTE: 1 - For WCDMA (FDD) The number of these fields depends on the number of physical channels (N_{pc}) and on the 2ms/10ms TTI selector as illustrated in Table 4-117.

Table 4-117. WCDMA (FDD) Number of Fields

2ms/10ms/ N_{pc}	1	2	4
10ms TTI	5	5	10
2ms TTI	6	6	12

Segment is defined as a set of PAM symbols with a unique noise scaling factor: it represents number of PAM symbols in one half slot (2ms TTI case) or the number of PAM symbols in a sub-frame (10ms TTI case).

NOTE: 2 - In the case when N_{pc} = 4 the size of sub-frames (2ms TTI case) or frames (10ms TTI case) #2 and #3 is exactly one half of the size of sub-frames/frames #0 and #1

4.7.3.40 CINIT_P2_CFG

Figure 4-131. Cinit P2 Value Config Register

31	30	0
Reserved	CINIT_P2	

Table 4-118. Cinit P2 Value Config Register Details

Bit	Name	Description
31	Reserved	Value = NA
30-0	cinit_p2	C_{init} value for descrambler P2 sequence generator

4.7.4 Error Bit Definitions

The SSL error bits are defined in Table 4-119. These are the errors bits that will be recorded in the SSL data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-119. SSL Error Bit Definitions

BIT	NAME
6	Reserved
5	Reserved
4	Radio Standard and Modulation Mode error. Detected at least one of the following: • radio_standard equals 1, 6 or 7 • mod_mode_sel equals 6 or 7
3	Packet size error. Detected at least one of the following: • The packet payload has a size of zero, i.e., SOP & EOP & XCNT == 0. (All Radio Standards) • The packet payload has a size of 1 AND there are less than Valid bytes to process i.e. SOP & EOP & XCNT < Valid bytes. (All Radio Standards) – "Valid" number of bytes equals: LTE: 16 WiMax: 8 TDD CDMA: 4 FDD CDMA: 2 – Leftover data in the input buffer i.e. EOP has happened and all required data have been processed yet data still remains in the input buffer. This could also be thought of as early EOP. (All Radio Standards but LTE)
2	Invalid amount of input data. There was too much or too little data to process (All standards but WiMax). • LTE: Error if number of ingress symbol data != cmux_sz * rmux_ln where cmux_sz = 9, 10 11 or 12. • TD-SCDMA: Error if number of ingress symbol data != SUM of tdd_slot_size[tdd_num_slots] • FDD WCDMA: Error if number of ingress symbol data != – 10ms: 1 phy_ch: 5 * fdd_symbols_seg 2 phy_ch: 10 * fdd_symbols_seg 4 phy_ch: 15 * fdd_symbols_seg – 2ms: 1 phy_ch: 6 * fdd_symbols_seg 2 phy_ch: 12 * fdd_symbols_seg 4 phy_ch: 18 * fdd_symbols_seg
1	Noise Factor IEEE Format error. Detected at least one of the following: • Noise Factor sign is negative. (All Radio Standards) • Noise Factor is not a number. (All Radio Standards)
0	Out of range error. Detected at least one of the following: • Invalid Unit Value i.e. unit value <= 0. (All Radio Standards) • Invalid B matrix select i.e b2f_ssl_b_matrix_sel > 71. (LTE Only) • Invalid R'mux length index i.e. b2f_ssl_rmux_ln_index > 34. (LTE Only) • Invalid Number of Slots i.e. b2f_ssl_cdma_num_slots equal 0 or 7. (TDD CDMA Only) • Invalid Slot Size i.e b2f_ssl_cdma_size_slot_n == 0; for 0 < n <= b2f_ssl_cdma_num_slots. (TD-SCDMA Only) • Invalid Number of Physical Channels i.e. b2f_ssl_cdma_num_phy_ch equal to 0, 3, 5 or 7. (FDD WCDMA Only) • Invalid Number of Symbols per Segment i.e. b2f_ssl_cdma_symb_seg == 0. (FDD CDMA Only)

4.8 RD

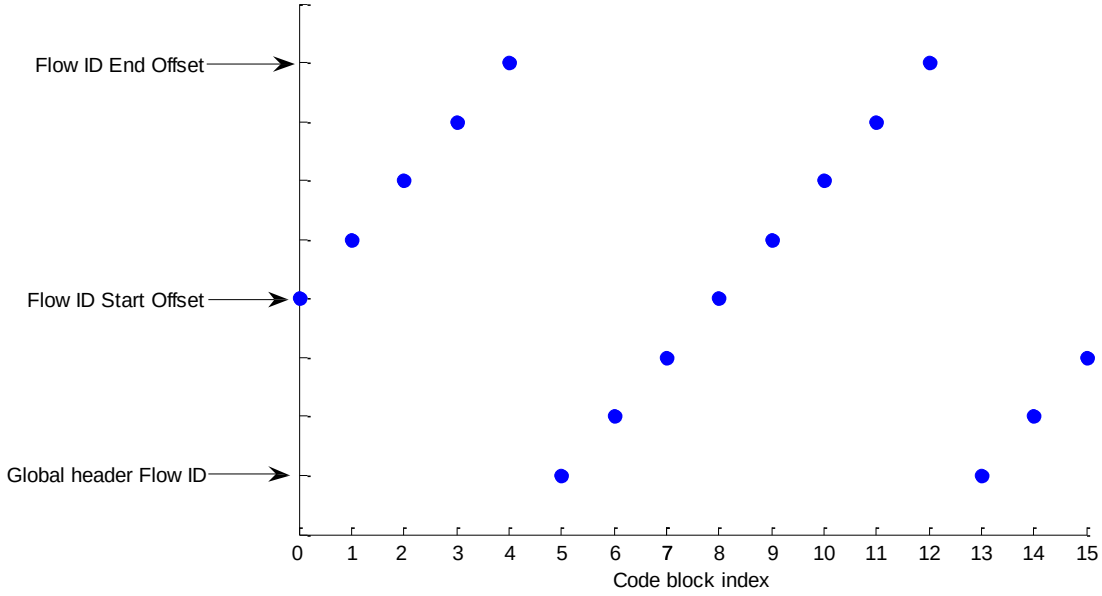
The RD submodule performs rate de-matching for all the standards supported by the BCP. It also supports code block segmentation, HARQ LLR combining (LLR history data stored/retrieved via embedded DIO module), and initial beta state calculation required by the TCP3d Turbo decoder module. The format of the input data is exactly like the output format of the RM submodule except that the inputs are always soft bits. HARQ history data is read and written via the embedded DIO module and is maintained at 8-bit precision. RD result data is output via the Packet DMA and has a different precision for different types of data. Code block data to be decoded has a precision of either 6-bit or 8-bit, depending on the turboDecDynRng config header parameter. It is usually 6-bit precision for Turbo encoded data (going to TCP3d) and 8-bit precision for convolutionally encoded data. The LTE CQI data always has 8-bit precision. The output code block data is also scaled by a provided scale factor. Please note that the 6-bit output data is actually sign-extended to 8 bits to ease debugging (the TCP3d only uses the least significant 6 bits). This means that the TCP3d must be configured to disable its error reporting that checks if the two upper bits are 0.

The RD submodule implements the “one-to-many” feature (for non-Rel-99 modes), generating a packet for each output code block. When using the RD submodule, as with all BCP packets, there is a flow_ID to identify the output flow. Specific to RD, every output code block can be routed to one of up to eight different flows using the following algorithm. The **flowIdStartingOffset** and **flowIdEndingOffset** are parameters in the RD header, each in the range of 0 to 7. The flow_ID used is then calculated as follows:

flow_ID_used(i) = (i + flowIdStartingOffset) mod (flowIdEndingOffset + 1) + globalHeader_flow_ID, for i = 0,1,2,… where i is the index of the output code block.

An example sequence for **flow_ID_used** is shown in Figure 4-132.

Figure 4-132. Example of the Packet Flow ID Used (one code block per packet) at the Output of RD



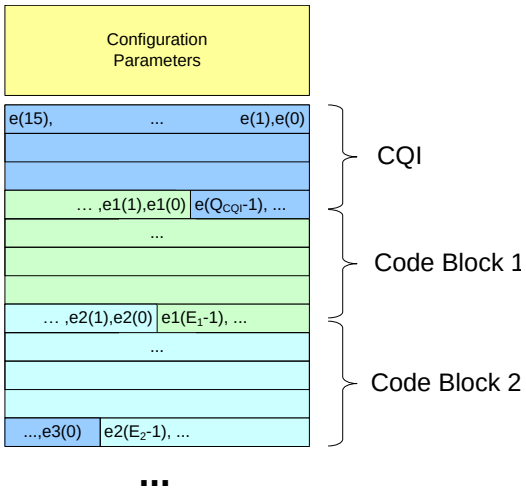
The input and output data formats as well as packet configuration fields differ by standard and are described in the following sections.

4.8.1 Data Format

4.8.1.1 LTE - Input Data

The RD input data for LTE is made of soft bits (one per byte) belonging to a single transport block, usually coming from the soft slicer (SSL) submodule. They are aligned and packed as shown in Figure 4-133.

Figure 4-133. RD Input Format for LTE



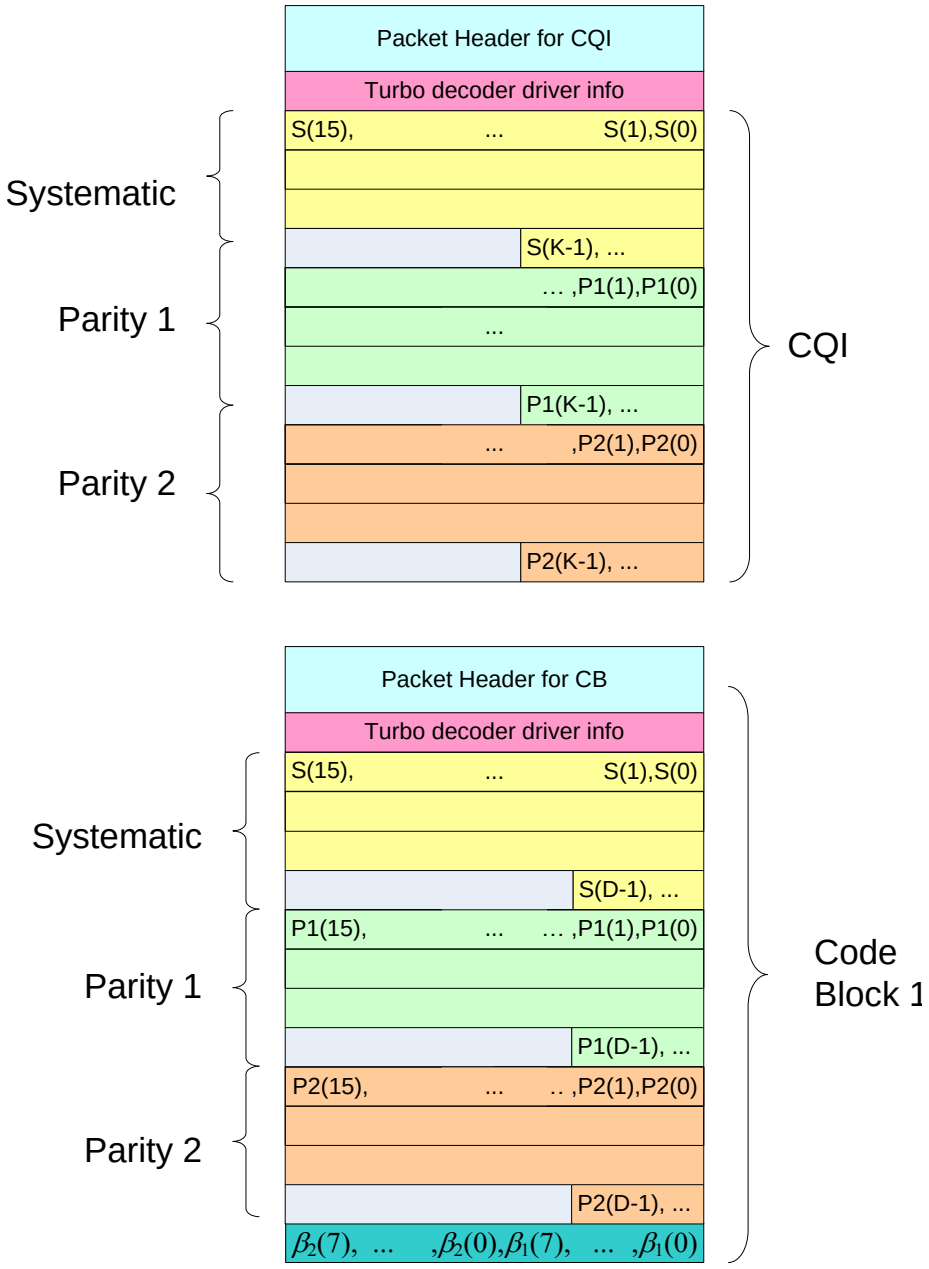
Please note that the input may contain only CQI, only code blocks, or both.

4.8.1.2 LTE - Output Data

The RD submodule outputs data to three destinations: 1) CQI data to DSP memory for the DSP, 2) code blocks to DSP memory for Turbo decoding, 3) HARQ history data to DSP memory for HARQ data storage. In the first two path s, the input transport block packet is split into multiple output packets send via the Packet DMA, one packet for the CQI data and one packet per code block. In the HARQ history data path, the code block data stays as one transport block and is transferred via the embedded DIO module. If the configuration parameter **enableHarqOutput** = 0, the saving of the whole transport block to HARQ memory is skipped.

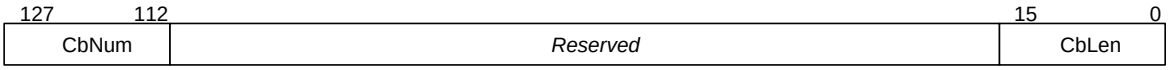
The output data format for CQI and Turbo decoding is shown in Figure 4-134. If the input transport block contains CQI data, the first output packet contains only the CQI data (rate de-matched if selected – when the size of CQI payload is greater than 11). The remaining packets contain the rate de-matched data per code block, with three streams always including tail bits (D=K+4), followed by a 16-byte array with the initial beta states required for the TCP3d Turbo decoder. The initial beta state calculation is described in the TCP3d User Guide.

Figure 4-134. RD Output Format for CQI and TCP3d for LTE



The first quad-word of each output packet contains information about code block size and the code block index within the transport block as shown in Figure 4-135. This information is used by the application to control the Turbo decoding. Typically, the different code blocks are distributed between several different TCP3d modules. The CBNum index can be used to ensure that the order of the code blocks is maintained after they are decoded by the different TCP3d modules.

Figure 4-135. Turbo Decoder Driver Information

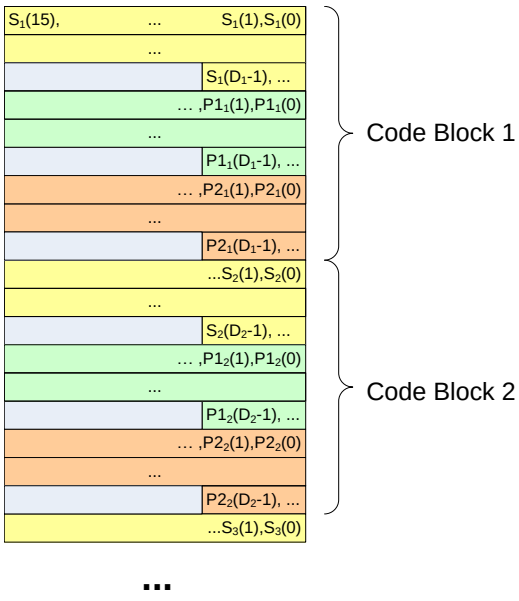


The fields are:

- CbNum:
  - CQI packet: 0xffff
  - Normal packet: Index number indicating code block number within transport block (starting with 0)
- CbLen: Output block length

The output data format for the HARQ memory is shown in Figure 4-136. The output transport block does not include the CQI data , packet header, turbo decoder driver info, or initial beta states. In LTE, the three code block streams include tail bits (D=K+4).

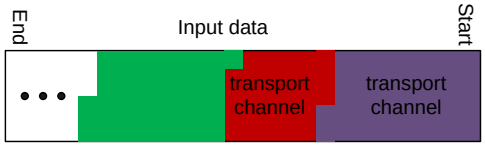
Figure 4-136. RD Output Format for HARQ for LTE



4.8.1.3 WCDMA / TD-SCDMA - Input Data

The RD input data is formatted as soft bits (one per byte). The input packet to the RD module consists of one or more transport channels (TrCHs). When multiple TrCHs exist in a packet, they are packed into a single continuous stream with no padding between them. The first input bit is bit 0 of the first 128-bit quad word of the payload. TrCHs can be any number of bits, although packets with a 0-length payload are not expected and will not be processed. The basic input packet structure is shown in Figure 4-137, where the start of the data is the right side.

Figure 4-137. RD Input Data Structure for WCDMA / TD-SCDMA

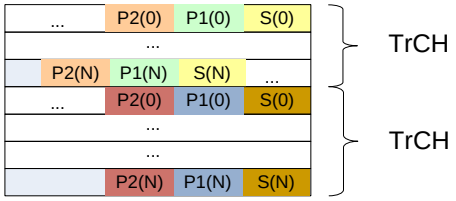


For Rel-99, up to 6 TrCHs are supported per packet, any two of which can be Turbo channels. For HSUPA, a single TrCH per packet is supported. HSUPA TrCHs must use the Sys0 channel and P0 parity set (header configuration 32-bit words 5 – 22).

4.8.1.4 WCDMA / TD-SCDMA - Output Data

The RD output data is formatted as soft bits (one per byte). The first output bit is bit 0 of the first 128-bit quad word of the payload. When multiple TrCHs exist in a packet for Rel-99, 0 padding is inserted as needed, such that each output TrCH begins on a 128-bit boundary, as shown in Figure 4-138.

Figure 4-138. RD Output Data Structure for Rel-99 (WCDMA / TD-SCDMA)

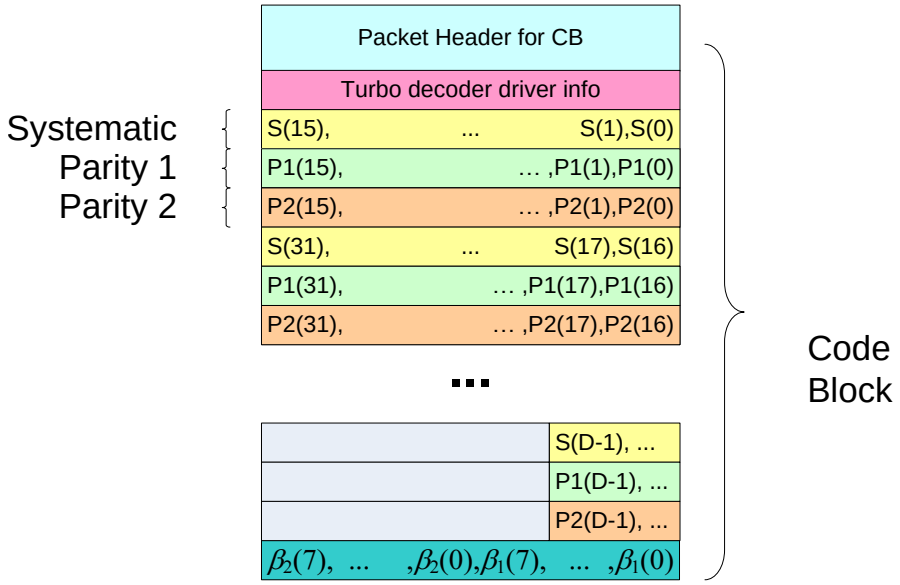


The RD can send HSUPA output data to two destinations:

- to DSP memory for Turbo decoding
- to DSP memory for HARQhistory data storage

The output data for Turbo decoding for HSUPA packets (which always contain a single TrCH) are segmented into one or more code blocks, each transmitted as an individual packet, as shown in Figure 4-139. Groups of 16 bytes of systematic, parity1 and parity2 soft bits are interlaced and then each stream is padded with 0s at the end if needed to form a multiple of 128 bits. They are then followed by a 16-byte array with the initial beta states required for the TCP3d Turbo decoder.

Figure 4-139. RD Output Format for Turbo Decoding for HSUPA (WCDMA / TD-SCDMA)



The first quad-word of each output HSUPA packet contains information about code block size and the code block index within the transport block as shown in Figure 4-140. This information is used by the turbo decoder driver.

Figure 4-140. Cinit P2 Value Config Register

127 112	111 16	15 0
CbNum	Reserved	CbLen

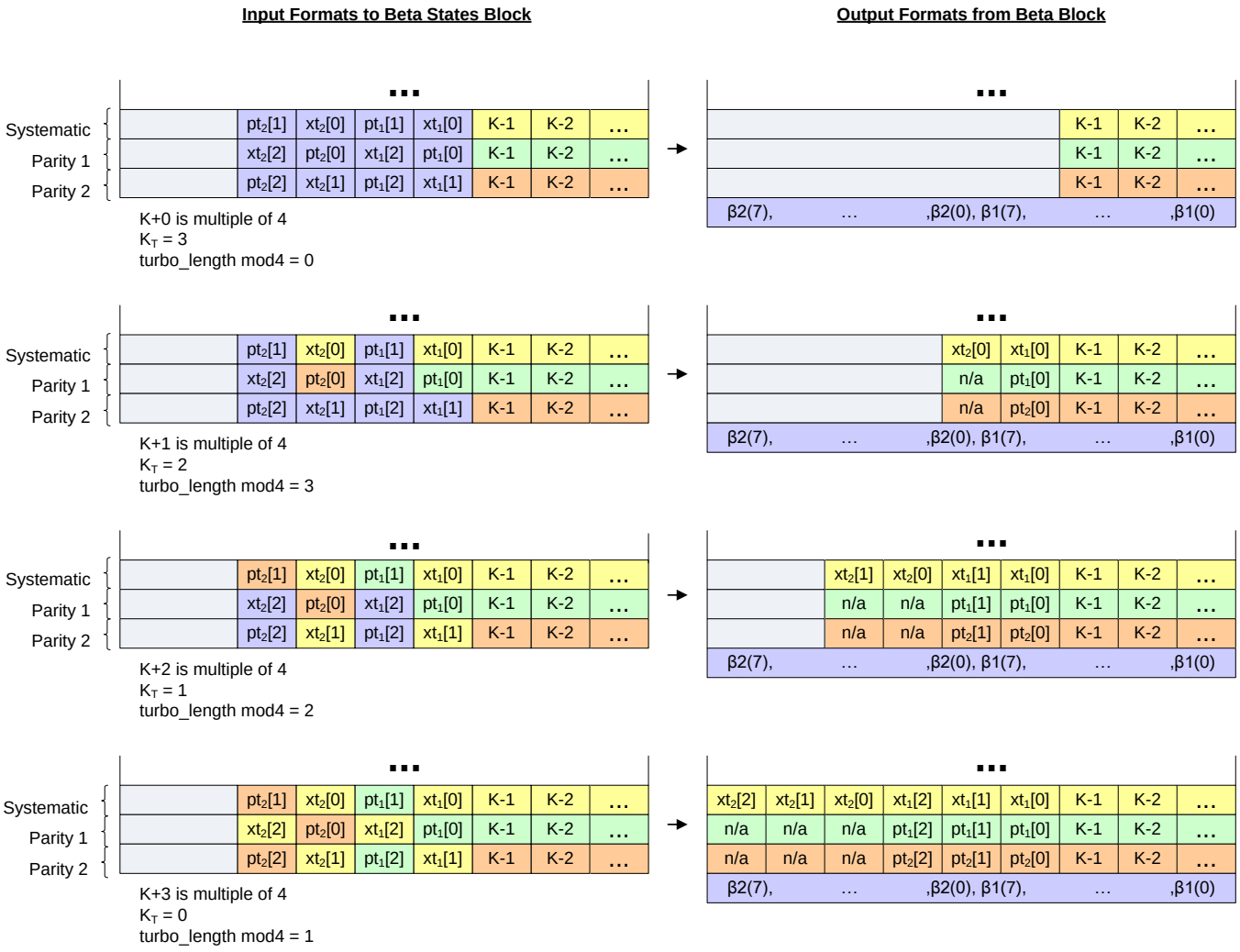
The fields are:

- CbNum: Index number indicating code block number within transport block (starting with 0)
- CbLen: Output block length

Since the TCP3d requires that the number of tail bits and their ordering be modified based upon the number of bits (K) in the code block, the RD adjusts its output accordingly. The value of the **turbo_length** configuration field still includes the input tail bits, and is therefore equal to K+4.

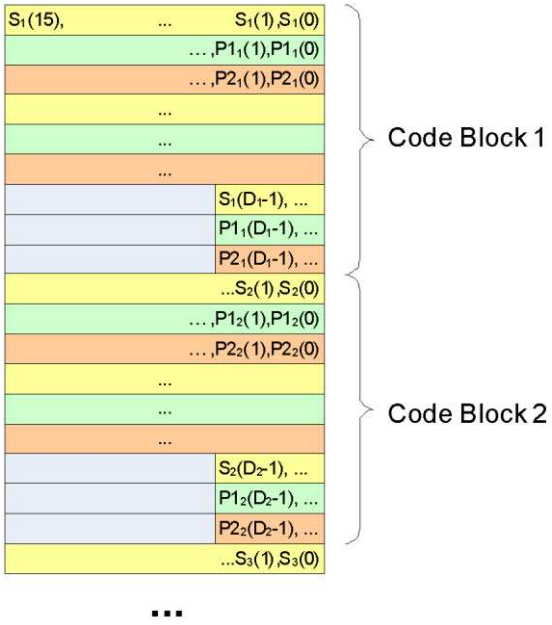
Figure 4-141 shows the mapping for the four possible tail bit reordering schemes. The left half of the figure shows the code block input formats and the right half of the figure shows the corresponding code block output formats sent to the TCP3d. Note that the input length and output length are different for 3 of the four modes. The n/a bits are dummy bits (don't care values). The left half of the figure also indicates which input tail bits are used to calculate the beta states (purple bits). The initial beta state calculation is described in the TCP3d User Guide.

Figure 4-141. RD HSUPA Tail Bit Re-Ordering



The format of the HSUPA output data for the HARQ memory is the same as the data for Turbo decoding, except there is no packet header, Turbo decoder driver info , or initial beta states, and the entire transport block is sent in a singl e data block. Also, the HARQ data is always 8-bit precision, whereas the data for Turbo decoding has a programmable precision (controlled by the turboDecDynRng config header field), but should be set to 6-bit for TCP3d. Please note that the 6-bit data for TCP3d is actually sign-extended to 8 bits to ease debugging (the TCP3d only uses the least significant 6 bits). This means that the TCP3d must be configured to disable its error reporting that checks if the two upper bits are 0. The HARQ data format is shown in Figure 4-142.

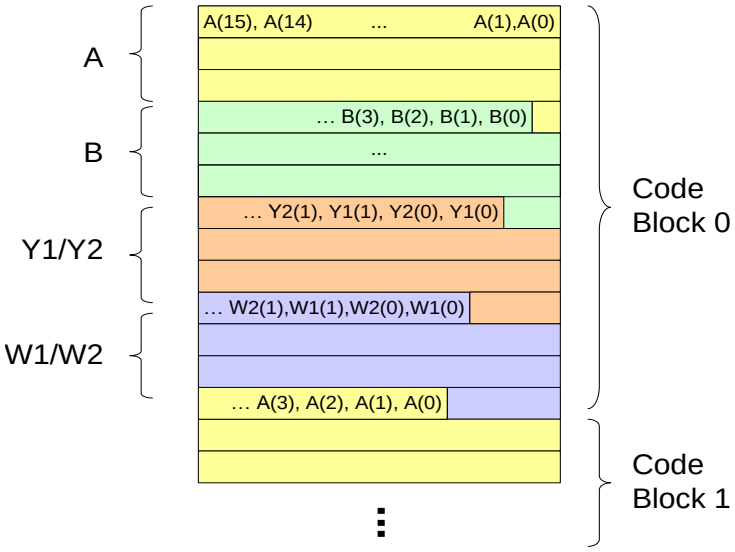
Figure 4-142. RD Output Format for HARQ for HSUPA



4.8.1.5 WiMAX - Input Data

For WiMAX, the input data is the same as LTE (one soft bit per byte) except there are no CQI bits and instead of three streams for S, P1, P2 , there are four portions of the data for WiMAX: A , B, Y1 /W1, and Y2 /W2, in that order. The 'X/Y' means that those soft bits are interleaved so they alternate after each soft bit. This is illustrated in Figure 4-143. Any input data that is not present will be populated with 0s.

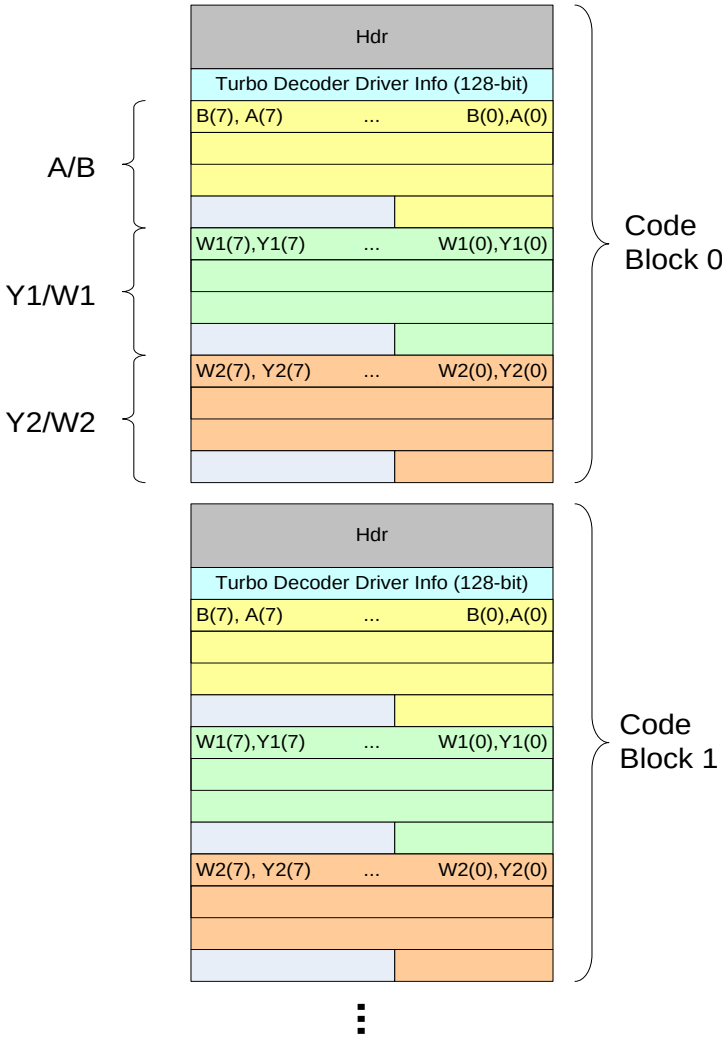
Figure 4-143. RD Input Format for WiMAX



4.8.1.6 WiMAX - Output Data

For WiMAX, the output data is similar to LTE mode (one soft bit per byte) except there are no CQI bits, no tail bits (D = K), and the initial beta states are not present. The output for Turbo decoding is shown in Figure 4-144.

Figure 4-144. RD Output Format for TCP3d for WiMAX



The output for HARQ is the same as the output for TCP3d except the re is no header or Turbo decoder driver info. Also, the HARQ data is 8-bit precision, whereas the data for TCP3d is usually 6-bit. Please note that the 6-bit data for TCP3d is actually sign-extended to 8 bits to ease debugging (the TCP3d only uses the least significant 6 bits). This means that the TCP3d must be configured to disable its error reporting that checks if the two upper bits are 0.

4.8.2 Memory Mapped Registers

The Rate De-matcher DIO VBUSM Priority register is defined at the first address of the RD memory map, at address BCP base address + 0x0B00.

Figure 4-145. RD DIO VBUSM PRIORITY Register

31	3	2	0
Reserved		PRIORITY	
R		R/W - 0	

Legend: R = Read only; W = Write only

Table 4-120. RD DIO VBUSM PRIORITY Register Details

Bits	Name	Description
31-3	Reserved	Reserved
2-0	Priority	0 = highest, 7 = lowest reset value = 0 (highest)

4.8.2.1 RD Data Saturation Registers (Available in Keystone-II Devices)

The starting offset of the RD sub-module relative to BCP base address is 0x0B00. The register address offsets in the table below are relative to RD base address.

Table 4-121. RD Data Saturation Registers

Name	Bits	Address Offset	Access	Reset	Description
RD_LTE_LLRSATVALUES	32	0x0004	R/W	0xFFFF7F1F	LLR saturation values in the LTE RD module
RD_WIMAX_LLRSATVALUES	32	0x0008	R/W	0x00007F1F	LLR saturation values in the WiMAX RD module
RD_HSPA_LLRSATVALUES	32	0x000C	R/W	0xFFFF7F1F	LLR saturation values in the HSPA RD module
RD_R99_LLRSATVALUES	32	0x0010	R/W	0x00007F1F	LLR saturation values in the Rel-99 RD module

Figure 4-146. RD_LTE_LLRSATVALUES Register

31 24	23 16	15 8	7 0
lte_llrTailSatValue_1	lte_llrTailSatValue_0	lte_llrDataSatValue_1	lte_llrDataSatValue_0

Table 4-122. RD_LTE_LLRSATVALUES Register Details

Bits	Name	R/W	Reset	Description
31-24	lte_llrTailSatValue_1	R/W	255	Tail LLRs towards Beta States Block are saturated to +/- lte_llrTailSatValue_1 when turboDecDynRng=1
23-16	lte_llrTailSatValue_0	R/W	255	Tail LLRs towards Beta States Block are saturated to +/- lte_llrTailSatValue_0 when turboDecDynRng=0
15-8	lte_llrDataSatValue_1	R/W	127	All LLRs towards TCP3d are saturated to +/- lte_llrDataSatValue_1 when turboDecDynRng=1
7-0	lte_llrDataSatValue_0	R/W	31	All LLRs towards TCP3d are saturated to +/- lte_llrDataSatValue_0 when turboDecDynRng=0

Figure 4-147. RD_WIMAX_LLRSATVALUES Register

31 16	15 8	7 0
Reserved	wimax_llrDataSatValue_1	wimax_llrDataSatValue_0

Table 4-123. RD_WIMAX_LLRSATVALUES Register Details

Bits	Name	R/W	Reset	Description
31-16	Reserved	R/W	0	Reserved
15-8	wimax_llrDataSatValue_1	R/W	127	All LLRs towards TCP3d are saturated to +/- wimax_llrDataSatValue_1 when turboDecDynRng=1
7-0	wimax_llrDataSatValue_0	R/W	31	All LLRs towards TCP3d are saturated to +/- wimax_llrDataSatValue_0 when turboDecDynRng=0

Figure 4-148. RD_HSPA_LLRSATVALUES Register

31 24	23 16	15 8	7 0
hspa_llrTailSatValue_1	hspa_llrTailSatValue_0	hspa_llrDataSatValue_1	hspa_llrDataSatValue_0

Table 4-124. RD_HSPA_LLRSATVALUES Register Details

Bits	Name	R/W	Reset	Description
31-24	hspa_llrTailSatValue_1	R/W	255	Tail LLRs towards Beta States Block are saturated to +/- hspa_llrTailSatValue_1 when turboDecDynRng=1
23-16	hspa_llrTailSatValue_0	R/W	255	Tail LLRs towards Beta States Block are saturated to +/- hspa_llrTailSatValue_0 when turboDecDynRng=0
15-8	hspa_llrDataSatValue_1	R/W	127	All LLRs towards TCP3d are saturated to +/- hspa_llrDataSatValue_1 when turboDecDynRng=1
7-0	hspa_llrDataSatValue_0	R/W	31	All LLRs towards TCP3d are saturated to +/- hspa_llrDataSatValue_0 when turboDecDynRng=0

Figure 4-149. RD_R99_LLRSATVALUES Register

31 16	15 8	7 0
Reserved	r99_llrDataSatValue_1	r99_llrDataSatValue_0

Table 4-125. RD_R99_LLRSATVALUES Register Details

Bits	Name	R/W	Reset	Description
31-16	Reserved	R/W	0	Reserved
15-8	r99_llrDataSatValue_1	R/W	127	All LLRs towards DSP are saturated to +/- r99_llrDataSatValue_1 when turboDecDynRng=1
7-0	r99_llrDataSatValue_0	R/W	31	All LLRs towards DSP are saturated to +/- r99_llrDataSatValue_0 when turboDecDynRng=0

4.8.3 Packet Header Configuration Fields - LTE

4.8.3.1 RD_LTE_HDR_CFG0

Figure 4-150. RD LTE Header Config Word 0

31 29	28 24	23 20	19	18	17
Reserved	TCP3DSCALEFACTOR	Reserved	TURBODECDYNRNG	ENABLEHARQOUTPUT	ENABLEHARQINPUT

16 12	11 8	7	6 0
Reserved	MOD_ID	Rsvd	LOCAL_HDR_LEN

Table 4-126. RD LTE Header Config Word 0 Details

Bit	Name	Description
31-29	Reserved	Reserved
28-24	TCP3DSCALEFACTOR	Value = 0 - 31 Unsigned scale factor in Q4 format, so that the value 1 is represented as 16.
23-20	Reserved	Reserved
19	TURBODECDYNRNG	• 0: LLRs towards turbo decoder are in range +/-31 (use for normal operation) • 1: LLRs towards turbo decoder are in range +/-127 (for debug only) Note that when '0' is selected, 6-bit data is actually sign-extended to 8-bit for easier debug (TCP3d does not use upper two bits).
18	ENABLEHARQOUTPUT	Enable HARQ output flag. • 0 – do not write back updated history data. • 1 – write back the whole TB to history data.
17	ENABLEHARQINPUT	Enable HARQ input flag. • 0 – do not use history data in LLR combining • 1 – use history data.
16-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

4.8.3.2 RD_LTE_HDR_CFG1

Figure 4-151. RD LTE Header Config Word 1

31 24	23 21	20 18	17 15
Reserved	FLOWIDCQIOFFSET	FLOWIDHI	INITCBFLOWID

14 13	12 7	6 1	0
Reserved	NUMFILLERBITSF	RVSTARTCOLUMN	CQIPASSEDTHROUGH

Table 4-127. RD LTE Header Config Word 1 Details

Bit	Name	Description
31-24	Reserved	Reserved
23-21	flowIdCqiOffset	Value = 0 - 7 Offset for CQI CB only. Added to FlowId in PPB. FlowId is a Global Header field.
20-18	flowIdHi	Value = 0 - 7 Max value for flow id counter, Max value prior to wrapping back to 0. Flow id counter is added to baseFlowId in PPB yielding the actual flow_id to be used.
17-15	InitCbFlowId	Value = 0 - 7 Starting value for flow id counter (for first non-CQI CB in TB). Value to be added to baseFlowId in PPB.

Table 4-127. RD LTE Header Config Word 1 Details (continued)

Bit	Name	Description
14-13	Reserved	Reserved
12-7	numFillerBitsF	Value = 0 - 63 Number of filler bits F in the first code block.

Table 4-127. RD LTE Header Config Word 1 Details (continued)

Bit	Name	Description
6-1	rvStartColumn	Value = 0 - 63 Starting column in the writing phase to dual matrix. The writing starts always from the top of the column. This is same far all CBs. CQI packet assumes rvindex starting at row0, col0
0	cqiPassedThrough	Control data flag • 0 – CQI bits are rate dematched. • 1 – bypass rate dematching of the CQI data.

4.8.3.3 RD_LTE_HDR_CFG2

Figure 4-152. RD LTE Header Config Word 2

31 24	23 16	15 8	7 0
NUMCODEBLOCKSCE2	NUMCODEBLOCKSC2	NUMCODEBLOCKSCE1	NUMCODEBLOCKSC1

Table 4-128. RD LTE Header Config Word 2 Details

Bit	Name	Description
31-24	numCodeBlocksCe2	Value = 0 - 255 Input: Number of code blocks C_{e2} with transmitted size E_2
23-16	numCodeBlocksC2	Value = 0 - 255 Output: Number of code blocks C_2 of size K_2
15-8	numCodeBlocksCe1	Value = 0 - 255 Input: Number of code blocks C_{e1} with transmitted size E_1. (Total number of CBs in TB is $C = C_{e1} + C_{e2}$)
7-0	numCodeBlocksC1	Value = 0 - 255 Output: Number of code blocks C_1 of size K_1. (Total number of CBs in TB is $C = C_1 + C_2$)

4.8.3.4 RD_LTE_HDR_CFG3

Figure 4-153. RD LTE Header Config Word 3

31 16	15 14	13 0
BLOCKSIZEE1	Reserved	BLOCKSIZEK1

Table 4-129. RD LTE Header Config Word 3 Details

Bit	Name	Description
31-16	blockSizeE1	Value = <65k Input: Transmitted size E_1, $E_r = E1$, for r=0,…, C_{e1}-1 (the size at the input side to RD module).
15-14	Reserved	Value = NA
13-0	blockSizeK1	Value = 40-8k Output: Code block size K_1, $K_r = K_1$ for $_r$ =0,…, C_1-1 The size at the output of RD module is 3x(K_r+4) Note: actual supported size for non-CQI is 8K including tail bits and padding.

4.8.3.5 RD_LTE_HDR_CFG4

Figure 4-154. RD LTE Header Config Word 4

31 16	15 14	13 0
BLOCKSIZEE2	Reserved	BLOCKSIZEK2

Table 4-130. RD LTE Header Config Word 4 Details

Bit	Name	Description
31-16	blockSizeE2	Value = <65k Input: Transmitted size E_2, $E_r = E_2$, for $r = C_{e1},\ldots, C_{e1} + C_{e2}-1$
15-14	Reserved	Value = NA
13-0	blockSizeK2	Value = 40-8k Output: Code block size K_2, $K_r = K_2$ for $r = C_1,\ldots,(C_1 + C_2-1)$

4.8.3.6 RD_LTE_HDR_CFG5

Figure 4-155. RD LTE Header Config Word 5

31 16	15 14	13 0
BLOCKSIZEINQCQI	Reserved	BLOCKSIZEOUTQCQI

Table 4-131. RD LTE Header Config Word 5 Details

Bit	Name	Description
31-16	blockSizeInQcqi	Input: Transmitted size of CQI data. This is the input size to RD module.
15-14	Reserved	Value = NA
13-0	blockSizeOutQcqi	Output: Payload size of CQI data. If blockSizeOutQcqi = 0, the number of CQI data blocks (N_{CQI}) is 0. If blockSizeQcqi > 0, then NCQI = 1. The size at the output of RD module is 3x blockSizeOutQcqi, except when cqiPassedThrough=1 the output size is equal to blockSizeInQcqi.

4.8.3.7 RD_LTE_HDR_CFG6

Figure 4-156. RD LTE Header Config Word 6

31 0
HARQINPUTADDRESS

Table 4-132. RD LTE Header Config Word 6 Details

Bit	Name	Description
31-0	harqInputAddress	Address of the HARQ input data. The data length is C_1*3*floor((K_1+4+15)/16)*16+C_2*3*floor((K_2+4+15)/16)*16 bytes.

4.8.3.8 RD_LTE_HDR_CFG7

Figure 4-157. RD LTE Header Config Word 7

Table 4-133. RD LTE Header Config Word 7 Details

Bit	Name	Description
31-0	harqOutputAddress	Address of the HARQ output data. The data length is C_1*3*floor((K_1+4+15)/16)*16+ C_2*3*floor((K_2+4+15)/16)*16 bytes.

4.8.4 Packet Header Configuration Fields - WCDMA / TD-SCDMA

Figure 4-158 provides a WCDMA / TD-SCDMA packet header configuration summary.

Figure 4-158. WCDMA / TD-SCDMA Packet Header Configuration Summary

Channel Mapping:

PACKET CONFIG WORDS
CHANNEL 0
P0 PARITY 1
P0 PARITY 2
CHANNEL 1
CHANNEL 2
CHANNEL 3
CHANNEL 4
CHANNEL 5
P1 PARITY 1
P1 PARITY 2

Rate De-Match Configuration Data Format

QUAD WORD	WORD3	WORD2	WORD1	WORD0
0	harq_in_addr[31:0]	turbo_length[15:0] turbo_count[7:0] flow_id_init[2:0] flow_id_max[2:0] res[1:0]	collect_cols[23:0] collect_rows[2:0] res[4:0]	CommonHdr[15:0] PktCfgFlags[3:0] res[3:0] tcp3_scale[4:0] res[2:0]
1	minus2[23:0] res[7:0]	init2[23:0] res[7:0]	length[23:0] res[7:0]	harq_out_addr[31:0]
2	minus_1[23:0] res[7:0]	init_1[23:0] res[7:0]	length[23:0] res[7:0]	plus2[23:0] alpha[1:0] beta[1:0] puncture turbo[1:0] res
3	plus_2[23:0] res[7:0]	minus_2[23:0] res[7:0]	init_2[23:0] res[7:0]	plus_1[23:0] res[7:0]
4	plus_1[23:0] res[7:0]	minus_1[23:0] res[7:0]	init_1[23:0] res[7:0]	length[23:0] res[7:0]
5	length[15:0] init2[15:0]	plus_2[23:0] res[7:0]	minus_2[23:0] res[7:0]	init_2[23:0] res[7:0]
6	minus2[15:0] plus2[15:0]	length[15:0] init2[15:0]	alpha[1:0] beta[1:0] puncture turbo[1:0] res[24:0]	minus2[15:0] plus2[15:0]
7	alpha[1:0] beta[1:0] puncture turbo[1:0] res[24:0]	minus2[15:0] plus2[15:0]]	length[15:0] init2[15:0]	alpha[1:0] beta[1:0] puncture turbo[1:0] res[24:0]
8	length[15:0] init2[15:0]	alpha[1:0] beta[1:0] puncture turbo[1:0] res[24:0]	minus2[15:0] plus2[15:0]	length[15:0] init2[15:0]
9	minus2[15:0] plus2[15:0]]	length[15:0] init2[15:0]	alpha[1:0] beta[1:0] puncture turbo[1:0] res[24:0]	minus2[15:0] plus2[15:0]]
10			minus2[15:0] plus2[15:0]]	length[15:0] init2[15:0]

4.8.4.1 ppb_eng_cfg

Table 4-134. Packet Header Configuration Details (WCDMA/TD-SCDMA)

Word	Bits	Name	Description
0	31-29	Reserved	Reserved
	28-24	tcp3_scale	Value = any Scale factor that determines how LLR data is extracted from HARQ combined data for sending to the TCP3d
	23-20	Reserved	Reserved
	19	tcp3_dyn_rng	Tcp3d dynamic range select : • 0 = LLRs towards turbo decoder are in range +/-31 (use for normal operation) • 1 = LLRs towards turbo decoder are in range +/-127 (for debug only) Note that when '0' is selected, 6-bit data is actually sign-extended to 8-bit for easier debug (TCP3d does not use upper two bits).
	18	en_harq_out	Enable storing of history data for future LLR combining (HSUPA only, ignored when sys0_turbo field != 3) • 0 = Do not store history data • 1 = Store history data
	17	en_harq_in	Enable history data for LLR combining (HSUPA only, ignored when sys0_turbo field != 3) • 0 = Do not use history data • 1 = Use history data
	16	fdd	Un-collection method for HSUPA channels: • 0 = Perform HSPA un-collection • 1 = Perform Rel-99 un-collection
	15-12	Reserved	Reserved
	11-8	mod_id	Module ID for this submodule (see Table 4-1 table)
	7	Reserved	Reserved
	6-0	local_hdr_len	Local header length in 32-bit words, excluding this word (Word 0).
1	31-27	Reserved	Reserved
	26-24	collect_rows	Value = 2,4,6 Collect rows: Number of rows in bit collection table for HSUPA TD-SCDMA turbo channel
	23-0	collect_cols	Value = any Collect columns: Number of columns in bit collection table HSUPA TD-SCDMA turbo channel
2	31-30	Reserved	Reserved
	29-27	flow_id_max	Value = 0 - 7 Maximum value of replay_addr. The counter will wrap to 0 after this value. (HSUPA only, ignored when sys0_turbo field != 3)
	26-24	flow_id_init	Value = 0 - 7 Starting value of replay_addr. (value for first CB). This value is added to the base Flow ID used in the PPB. (HSUPA only, ignored when sys0_turbo field != 3)
	23-16	turbo_count	Value = any Number of turbo code blocks to output Used only for HSUPA channels
	15-0	turbo_length	Value = any
3	31-0	harq_in_addr	Value = any Starting address of HARQ history data input
4	31-0	harq_out_addr	Value = any Starting address of HARQ history data output
5	31-24	Reserved	Reserved
	23-0	sys0_len	Value = any Channel 0 length (in bits) after rate de-matching
6	31-24	Reserved	Reserved
	23-0	sys0_init2	Value = any Channel 0 initial value for 2nd rate-matching loop, and rate de-matching
7	31-24	Reserved	Reserved
	23-0	sys0_minus2	Value = any Channel 0 minus value for 2nd rate-matching loop, and rate de-matching

Table 4-134. Packet Header Configuration Details (WCDMA/TD-SCDMA) (continued)

Word	Bits	Name	Description
8	31	Reserved	Reserved
	30-29	sys0_turbo	Channel 0 channel type • 0 = Convolutional or Rel-99 Turbo Repeat • 1 = Rel-99 Turbo Puncture, use p0* parity parameters • 2 = Rel-99 Turbo Puncture, use p1* parity parameters • 3 = HSUPA, use p0* parity parameters
	28	sys0_puncture	Channel 0 Puncture flag: • 0 = Perform un-repeat during rate de-matching • 1 = Perform un-puncture during rate de-matching
	27-26	sys0_beta	Channel 0 beta value for Rel-99 bit separation
	25-24	sys0_alpha	Channel 0 alpha value for Rel-99 bit separation: • 0 = Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 = Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
	23-0	sys0_plus2	Value = any Channel 0 plus value for 2nd rate-matching loop, and rate de-matching
9	31-24	Reserved	Reserved
	23-0	p0_par1_len	Value = any P0 Parity1 length (in bits) after rate de-matching
10	31-24	Reserved	Reserved
	23-0	p0_par1_init1	Value = any P0 Parity1 initial value for 1st rate-matching loop (not used)
11	31-24	Reserved	Reserved
	23-0	p0_par1_minus1	Value = any P0 Parity1 minus value for 1st rate-matching loop (not used)
12	31-24	Reserved	Reserved
	23-0	p0_par1_plus1	Value = any P0 Parity1 plus value for 1st rate-matching loop (not used)
13	31-24	Reserved	Reserved
	23-0	p0_par1_init2	Value = any P0 Parity1 initial value for 2nd rate-matching loop, and rate de-matching
14	31-24	Reserved	Reserved
	23-0	p0_par1_minus2	Value = any P0 Parity1 minus value for 2nd rate-matching loop, and rate de-matching
15	31-24	Reserved	Reserved
	23-0	p0_par1_plus2	Value = any P0 Parity1 plus value for 2nd rate-matching loop, or and de-matching
16	31-24	Reserved	Reserved
	23-0	p0_par2_len	Value = any P0 Parity2 length (in bits) after rate de-matching
17	31-24	Reserved	Reserved
	23-0	p0_par2_init1	Value = any P0 Parity2 initial value for 1st rate-matching loop (not used)
18	31-24	Reserved	Reserved
	23-0	p0_par2_minus1	Value = any P0 Parity2 minus value for 1st rate-matching loop (not used)
19	31-24	Reserved	Reserved
	23-0	p0_par2_plus1	Value = any P0 Parity2 plus value for 1st rate-matching loop (not used)

Table 4-134. Packet Header Configuration Details (WCDMA/TD-SCDMA) (continued)

Word	Bits	Name	Description
20	31-24	Reserved	Reserved
	23-0	p0_par2_init2	Value = any P0 Parity2 initial value for 2nd rate-matching loop, and rate de-matching
21	31-24	Reserved	Reserved
	23-0	p0_par2_minus2	Value = any P0 Parity2 minus value for 2nd rate-matching loop, and rate de-matching
22	31-24	Reserved	Reserved
	23-0	p0_par2_plus2	Value = any P0 Parity2 plus value for 2nd rate-matching loop, and rate de-matching
23	31-16	sys1_init2	Value = any Channel 1 initial value for rate de-matching
	15-0	sys1_len	Value = any Channel 1 length (in bits) after rate de-matching
24	31-16	sys1_plus2	Value = any Channel 1 plus value for rate de-matching
	15-0	sys1_minus2	Value = any Channel 1 minus value for rate de-matching
25	31-7	Reserved	Reserved
	6-5	sys1_turbo	Channel 1 channel type • 0 = Convolutional or Rel-99 Turbo Repeat • 1 = Rel-99 Turbo Puncture, use p0* parity parameters • 2 = Rel-99 Turbo Puncture, use p1* parity parameters • 3 = Reserved, (hardware detects an error)
	4	sys1_puncture	Channel 1 Puncture flag: • 0 = Perform un-repeat during rate de-matching • 1 = Perform un-puncture during rate de-matching
	3-2	sys1_beta	Value = 0,1,2 Channel 1 beta value for Rel-99 bit separation
	1-0	sys1_alpha	Channel 1 alpha value for Rel-99 bit separation: • 0 = Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 = Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
26	31-16	sys2_init2	Value = any Channel 2 initial value for rate de-matching
	15-0	sys2_len	Value = any Channel 2 length (in bits) after rate de-matching
27	31-16	sys2_plus2	Value = any Channel 2 plus value for rate de-matching
	15-0	sys2_minus2	Value = any Channel 2 minus value for rate de-matching
28	31-7	Reserved	Reserved
	6-5	sys2_turbo	Channel 2 channel type • 0 = Convolutional or Rel-99 Turbo Repeat • 1 = Rel-99 Turbo Puncture, use p0* parity parameters • 2 = Rel-99 Turbo Puncture, use p1* parity parameters • 3 = Reserved, (hardware detects an error)
	4	sys2_puncture	Channel 2 Puncture flag: • 0 = Perform un-repeat during rate de-matching • 1 = Perform un-puncture during rate de-matching
	3-2	sys2_beta	Value = 0,1,2 Channel 2 beta value for Rel-99 bit separation
	1-0	sys2_alpha	Channel 2 alpha value for Rel-99 bit separation: • 0 = Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 = Use α_{sys}, α_{p1}, α_{p2} = 0,2,1

Table 4-134. Packet Header Configuration Details (WCDMA/TD-SCDMA) (continued)

Word	Bits	Name	Description
29	31-16	sys3_init2	Value = any Channel 3 initial value for rate de-matching
	15-0	sys3_len	Value = any Channel 3 length (in bits) after rate de-matching
30	31-16	sys3_plus2	Value = any Channel 3 plus value for rate de-matching
	15-0	sys3_minus2	Value = any Channel 3 minus value for rate de-matching
31	31-7	Reserved	Reserved
	6-5	sys3_turbo	Channel 3 channel type • 0 = Convolutional or Rel-99 Turbo Repeat • 1 = Rel-99 Turbo Puncture, use p0* parity parameters • 2 = Rel-99 Turbo Puncture, use p1* parity parameters • 3 = Reserved, (hardware detects an error)
	4	sys3_puncture	Channel 3 Puncture flag: • 0 = Perform un-repeat during rate de-matching • 1 = Perform un-puncture during rate de-matching
	3-2	sys3_beta	Value = 0,1,2 Channel 3 beta value for Rel-99 bit separation
	1-0	sys3_alpha	Channel 3 alpha value for Rel-99 bit separation: • 0 = Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 = Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
32	31-16	sys4_init2	Value = any Channel 4 initial value for rate de-matching
	15-0	sys4_len	Value = any Channel 4 length (in bits) after rate de-matching
33	31-16	sys4_plus2	Value = any Channel 4 plus value for rate de-matching
	15-0	sys4_minus2	Value = any Channel 4 minus value for rate de-matching
34	31-7	Reserved	Reserved
	6-5	sys4_turbo	Channel 4 channel type • 0 = Convolutional or Rel-99 Turbo Repeat • 1 = Rel-99 Turbo Puncture, use p0* parity parameters • 2 = Rel-99 Turbo Puncture, use p1* parity parameters • 3 = Reserved, (hardware detects an error)
	4	sys4_puncture	Channel 4 Puncture flag: • 0 = Perform un-repeat during rate de-matching • 1 = Perform un-puncture during rate de-matching
	3-2	sys4_beta	Value = 0,1,2 Channel 4 beta value for Rel-99 bit separation
	1-0	sys4_alpha	Channel 4 alpha value for Rel-99 bit separation: • 0 = Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 = Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
35	31-16	sys5_init2	Value = any Channel 5 initial value for rate de-matching
	15-0	sys5_len	Value = any Channel 5 length (in bits) after rate de-matching
36	31-16	sys5_plus2	Value = any Channel 5 plus value for rate de-matching
	15-0	sys5_minus2	Value = any Channel 5 minus value for rate de-matching

Table 4-134. Packet Header Configuration Details (WCDMA/TD-SCDMA) (continued)

Word	Bits	Name	Description
37	31-7	Reserved	Reserved
	6-5	sys5_turbo	Channel 5 channel type • 0 = Convolutional or Rel-99 Turbo Repeat • 1 = Rel-99 Turbo Puncture, use p0* parity parameters • 2 = Rel-99 Turbo Puncture, use p1* parity parameters • 3 = Reserved, (hardware detects an error)
	4	sys5_puncture	Channel 5 Puncture flag: • 0 = Perform un-repeat during rate de-matching • 1 = Perform un-puncture during rate de-matching
	3-2	sys5_beta	Value = 0,1,2 Channel 5 beta value for Rel-99 bit separation
	1-0	sys5_alpha	Channel 5 alpha value for Rel-99 bit separation: • 0 = Use α_{sys}, α_{p1}, α_{p2} = 0,1,2 • 1 = Use α_{sys}, α_{p1}, α_{p2} = 0,2,1
38	31-16	p1_par1_init2	Value = any P1 Parity1 initial value for 2nd rate-matching loop, and rate de-matching
	15-0	p1_par1_len	Value = any P1 Parity1 length (in bits) after rate de-matching
39	31-16	p1_par1_plus2	Value = any P1 Parity1 plus value for 2nd rate-matching loop, or and de-matching
	15-0	p1_par1_minus2	Value = any P1 Parity1 minus value for 2nd rate-matching loop, and rate de-matching
40	31-16	p1_par2_init2	Value = any P1 Parity2 initial value for 2nd rate-matching loop, and rate de-matching
	15-0	p1_par2_len	Value = any P1 Parity2 length (in bits) after rate de-matching
41	31-16	p1_par2_plus2	Value = any P1 Parity2 plus value for 2nd rate-matching loop, or and de-matching
	15-0	p1_par2_minus2	Value = any P1 Parity2 minus value for 2nd rate-matching loop, and rate de-matching

4.8.5 Packet Header Configuration Fields - WiMAX

4.8.5.1 RD_WIMAX_HDR_CFG0

Figure 4-159. RD WiMax Header Config Word 0

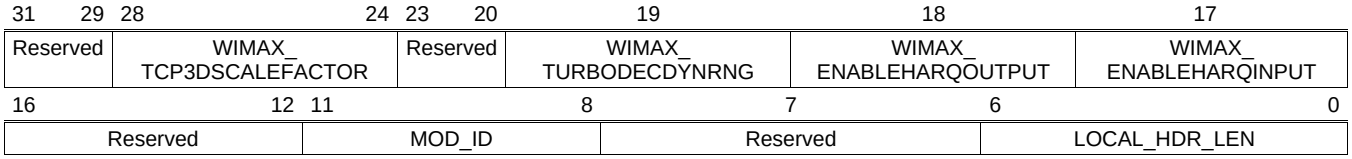


Table 4-135. RD WiMax Header Config Word 0 Details

Bit	Name	Description
31-29	Reserved	Reserved
28-24	WiMax_Tcp3dScaleFactor	Value = 0 - 31 Unsigned scale factor in Q4 format, so that the value 1 is represented as 16.
23-20	Reserved	Reserved
19	WiMax_turboDecDynRng	• 0: LLRs towards turbo decoder are in range +/-31 • 1: LLRs towards turbo decoder are in range +/-127
18	WiMax_enableHarqOutput	Enable HARQ output flag. • 0 – do not write back updated history data. • 1 – write back the whole TB to history data.
17	WiMax_enableHarqInput	Enable HARQ input flag. • 0 – do not use history data in LLR combining • 1 – use history data.
16-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

4.8.5.2 RD_WIMAX_HDR_CFG1

Figure 4-160. RD WiMax Header Config Word 1

31 30	29 16	15 13
Reserved	WIMAX_BLOCKSIZEK1	WIMAX_FLOWIDENDINGOFFSET

12 10	9	8 0
WIMAX_FLOWIDSTARTINGOFFSET	Reserved	WIMAX_NUMCODEBLOCKSC1

Table 4-136. RD WiMax Header Config Word 1 Details

Bit	Name	Description
31-30	Reserved	Value = NA
29-16	WiMax_blockSizeK1	Value = 48-8192 Code block size K_1, $K_r = K_1$ for r =0,…, C_1-1
15-13	WiMax_flowIdEndingOffset	Value = 0 - 7 The ending flow ID offset value (FLID_END).
12-10	WiMax_flowIdStartingOffset	Value = 0 - 7 The starting flow ID offset value (FLID_START) added to the packet flow ID value specified in the general header. The flow ID is incremented per code block, as FLID(i) = FLID_START + i mod (FLID_END-FLID_START+1), for i=0,1,2,3,… where i is the index of the output code block.
9	Reserved	Value = NA
8-0	WiMax_numCodeBlocksC1	Value = 0 - 511 Number of code blocks C_1 of size K_1. (Total number of CBs in Burst is $C = C_1 + C_2 + C_3$ where: $C_2 \begin{cases} 0 & K_2 & 0 \\ 1 & K_2 & 0 \end{cases}$ and $C_3 \begin{cases} 0 & K_3 & 0 \\ 1 & K_3 & 0 \end{cases}$

4.8.5.3 RD_WIMAX_HDR_CFG2

Figure 4-161. RD WiMax Header Config Word 2

31 16	15 7	6 4	3 0
WIMAX_BLOCKSIZEE1	Reserved	WIMAX_PARAM_J1	WIMAX_PARAM_M1

Table 4-137. RD WiMax Header Config Word 2 Details

Bit	Name	Description
31-16	WiMax_blockSizeE1	Value = <65k Transmitted size E_1, $E_r = E_1$, for r=0,…, C_1-1
15-7	Reserved	Value = NA
6-4	WiMax_param_J1	Value = ≤ 4 Sub-block interleaver parameter J for CBs of size K_1.
3-0	WiMax_param_m1	Value = ≤10 Sub-block interleaver parameter m for CBs of size K_1.

4.8.5.4 RD_WIMAX_HDR_CFG3

Figure 4-162. RD WiMax Header Config Word 3

31 30	29 16	15 2	1 0
Reserved	WIMAX_BLOCKSIZEK2	Reserved	WIMAX_NUMCODEBLOCKSC2

Table 4-138. RD WiMax Header Config Word 3 Details

Bit	Name	Description
31-30	Reserved	Value = NA
29-16	WiMax_blockSizeK2	Value = 48-8192 Code block size K_2, $K_r = K_2$ for $r = C_1$
15-2	Reserved	Value = NA
1-0	WiMax_numCodeBlocksC2	Value = 0-3 Number of code blocks C_2 of size K_2.

4.8.5.5 RD_WIMAX_HDR_CFG4

Figure 4-163. RD WiMax Header Config Word 4

31 16	15 7	6 4	3 0
WIMAX_BLOCKSIZEE2	Reserved	WIMAX_PARAM_J2	WIMAX_PARAM_M2

Table 4-139. RD WiMax Header Config Word 4 Details

Bit	Name	Description
31-16	WiMax_blockSizeE2	Value = <65k Transmitted size (E_2), $E_r = E_2$, for $r = C_1,\ldots, C_1 + C_2 - 1$
15-7	Reserved	Value = NA
6-4	WiMax_param_J2	Value = ≤ 4 Sub-block interleaver parameter J for CBs of size K_2.
3-0	WiMax_param_m2	Value = ≤10 Sub-block interleaver parameter m for CBs of size K_2.

4.8.5.6 RD_WIMAX_HDR_CFG5

Figure 4-164. RD WiMax Header Config Word 5

31 30	29 16	15 2	1 0
Reserved	WIMAX_BLOCKSIZEK3	Reserved	WIMAX_NUMCODEBLOCKSC3

Table 4-140. RD WiMax Header Config Word 5 Details

Bit	Name	Description
31-30	Reserved	Value = NA
29-16	WiMax_blockSizeK3	Value = 48-8192 Code block size K_3, $Kr = K_3$ for $r = C1+1$
15-2	Reserved	Value = NA
1-0	WiMax_numCodeBlocksC3	Value = 0-3 Number of code blocks C_3 of size K_3.

4.8.5.7 RD_WIMAX_HDR_CFG6

Figure 4-165. RD WiMax Header Config Word 6

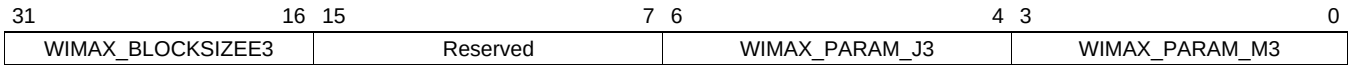


Table 4-141. RD WiMax Header Config Word 6 Details

Bit	Name	Description
31-16	WiMax_blockSizeE3	Value = <65k Transmitted size (E_3), $E_r = E_3$, for $r = C_2,\ldots, C_2 + C_3 - 1$
15-7	Reserved	Value = NA
6-4	WiMax_param_J3	Value = ≤ 4 Sub-block interleaver parameter J for CBs of size K_3.
3-0	WiMax_param_m3	Value = ≤10 Sub-block interleaver parameter m for CBs of size K_3.

4.8.5.8 RD_WIMAX_HDR_CFG7

Figure 4-166. RD WiMAx Header Config Word 7

Table 4-142. RD WiMax Header Config Word 7 Details

Bit	Name	Description
31-0	WiMax_ harqInputAddress	Address of the HARQ input data. The data length is C_1*3*floor((K_1+15)/16)*16+ C_2*3*floor((K_2+15)/16)*16+ C_3*3*floor((K_3+15)/16)*16 bytes.

4.8.5.9 RD_WIMAX_HDR_CFG8

Figure 4-167. RD WiMAx Header Config Word 8 Register

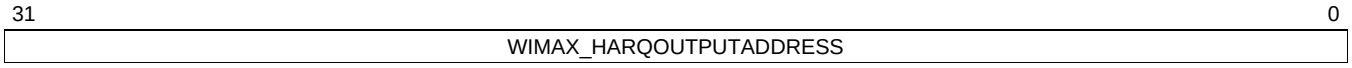


Table 4-143. RD WiMax Header Config Word 8 Details

Bit	Name	Description
31-0	WiMax_ harqOutputAddress	Address of the HARQ output data. The data length is C_1*3*floor((K_1+15)/16)*16+ C_2*3*floor((K_2+15)/16)*16+ C_3*3*floor((K_3+15)/16)*16 bytes.

4.8.6 Error Bit Definitions

The RD error bits are defined in Table 4-144. These are the errors bits that will be recorded in the RD data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-144. RD Error Bit Definitions

BIT	NAME
6	Internal HARQ DIO length mismatch with rate dematching length error.
5	Invalid packet header configuration.
4	Reserved
3	Reserved
2	Reserved
1	Payload data too big (late EOP).
0	Payload data too small (early EOP).

4.9 INT

WCDMA and TD-SCDMA define two interleavers, a first and second interleaver. Both are row to column interleavers, i.e., symbols are written in rows and read out in columns, taking into account inter-column permutation patterns as defined by the 3GPP specification. The first interleaver can have 1, 2, 4, or 8 columns (1 column means no interleaving); the second interleaver always has 30 columns. The size of the data symbol written/read and any packing or unpacking varies with the interleaver type. The second interleaver includes an additional function called constellation rearrangement which changes the bit ordering within a data symbol. The INT submodule performs all of this processing for the BCP, in addition to frame segmentation and DTX addition (for Rel-99).

Table 4-145 specifies which input and output format combinations are valid for the 1st interleaver (Rel-99).

Table 4-145. Valid INT Input/Output Format Combinations for 1st Interleaver (Rel-99)

numDataFormatIn →	1 bit per data value (=1)
numDataFormatOut ↓	
Rel-99 DL DTX (=0)	DTX_value ='1x'
1-bit packed (Rel-99 TDD) (=1)	DTX_value ='0'
2-bit packed (Rel-99 FDD) (=2)	DTX_value ='1x'

Table 4-146 specifies which input and output format combinations are valid for the 2nd interleaver (Rel-99 and HSPA). 2 = valid combination for 2nd interleaver, C = constellation mapping allowed, D = DTX allowed and specified in symbols.

Table 4-146. Valid INT Input/Output Format Combinations for 2nd Interleaver

numDataFormatIn >	1 bit per data value – BPSK (=1)	2 bits per data value – BPSK with DTX or 4PAM (=0)	2 bits per data value – QPSK (=2)	3 bits per data value – 8-PSK (=3)	4 bits per data value – 16-QAM (=4)	6 bits per data value – 64-QAM (=6)
numDataFormatOut ↓	Rel-99 TDD or HSUPA FDD	Rel-99 FDD (DTX) or HSUPA FDD (4PAM)	HSDPA (FDD and TDD), HSUPA TDD	HSPA TDD	HSDPA (FDD and TDD), HSUPA TDD	HSDPA (FDD and TDD)
1-bit packed (=1)	2D DTX='0'					
TAC format (FDD only) (=5)		2D DTX=TAC format	2		2C	2C
Unpacked – one data value per byte (=7)	2D DTX='00000000'	2D DTX='00000001x'	2	2	2C	2C

4.9.1 Data Format

4.9.1.1 Input Data

Each input data value may have 1, 2, 3, 4, or 6 input hard bits. Data values are stored starting in the least significant bit(s) of the first received 128-bit word and continue in a packed format. Three and six bit values may start in one 128-bit word and continue into the next 128-bit word. One, two, and four bit values are always aligned to 128-bit boundaries. When **NumFrameCount** is greater than one, there should not be any padding between input frames.

4.9.1.2 Output Data

Output data is stored in 128-bit quad-words. There are four possible output data formats. They are 1-bit packed, 2-bit packed, unpacked, and TAC format. The 1-bit and 2-bit packed modes pack every data value back-to-back with no gaps. That means there are 8 data values per byte for 1-bit packed format and 4 data values per byte for 2-bit packed format. When 2-bit packed data is used with DTX, the DTX bit will be the MS bit and the data bit the LS bit.

The unpacked data format is only used for the 2nd interleaver and places only one data value in each byte. DTX is represented as all 0s ('00000000'b) if the input was one bit per data value, or '0000001x'b (x = don't care) if the input was two bits per data value.

The TAC format is also only used for the 2nd interleaver. It matches the required input format for the Transmit Accelerator (TAC) block. Figure 4-168 shows the format for the least-significant 64 bits for Rel-99 FDD BPSK with DTX (2 bits per data value). This pattern repeats every 64 bits. Note that two BPSK symbols are combined into one QPSK symbol in the TAC, this corresponds to TAC Mode QPSK+DTX.

Figure 4-168. TAC Format for INT Output for BPSK with DTX

63	62	61	60		35	34	33	32
DTX_{I0}	DTX_{Q0}	DTX_{I1}	DTX_{Q1}	...	DTX_{I14}	DTX_{Q14}	DTR_{I15}	DTX_{Q15}
31	30	29	28		3	2	1	0
I_0	Q_0	I_1	Q_1	...	I_{14}	Q_{14}	I_{15}	Q_{15}

Figure 4-169 shows the format for the least-significant 64 bits for HSPA QPSK (TAC mode QPSK) or HSPA 16-QAM (TAC mode 16-QAM). This pattern repeats every 64 bits.

Figure 4-169. TAC Format for INT Output for QPSK or 16-QAM

63	62	61	60		35	34	33	32	31	30	29	28		3	2	1	0
I_{16}	Q_{16}	I_{17}	Q_{17}	...	I_{30}	Q_{30}	I_{31}	Q_{31}	I_0	Q_0	I_1	Q_1	...	I_{14}	Q_{14}	I_{15}	Q_{15}

Figure 4-170 shows the format for the least-significant 64 bits for HSPA 64-QAM (TAC mode 64-QAM). Note that there is a 16 bit gap of undefined bits from bit 47 to bit 32. This pattern repeats every 64 bits.

Figure 4-170. TAC Format for INT Output for 64-QAM

63	62	61	60		49	48	47		32	31	30	29	28		3	2	1	0
I_{16}	Q_{16}	I_{17}	Q_{17}	...	I_{23}	Q_{23}	X	...	X	I_0	Q_0	I_1	Q_1	...	I_{14}	Q_{14}	I_{15}	Q_{15}

When **NumFrameCount** > 1, each output frame will be separately aligned to start on a 128-bit word by 0 padding at the end of each frame. This padding allows the interleaver to also perform physical channel mapping on 128-bit aligned frames. When performing 1, 2, 4, or 8-way interleaving each output column is individually aligned to start at bit 0 of a 128-bit boundary. When performing 30-way interleaving the individual columns are packed together starting at the next unused bit of the last 128-bit word.

4.9.2 Memory Mapped Registers

The only memory mapped registers are the data logger registers and are described in the data logger section.

4.9.3 Packet Header Configuration Fields

Figure 4-171. INT Local Packet Header Word 0

31 12	11 8	7	6 0
Reserved	MOD_ID	Reserved	LOCAL_HDR_LEN

Table 4-147. INT Local Packet Header Word 0 Details

Bit	Name	Description
31-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

Figure 4-172. INT Local Packet Header Word 1

31 21	20 16	15 13	12 8	7
Reserved	NUM_INT_WAYS	Reserved	NUM_FRAME_COUNT	Reserved

6 4	3	2	1 0
ENUM_DATA_FORMAT_OUT	FLAG_HALF_RATE	FLAG_IN_ORDER	NUM_CONSTELLATION

Table 4-148. INT Local Packet Header Word 1 Details

Bits	Field Name	Description
31-21	Reserved	Reserved
20-16	NUM_INT_WAYS	1, 2, 4 or 8 selects 1st interleaving operation and refers to number of columns in matrix. 30 selects 2nd interleaving operation and uses 30 columns.
15-13	Reserved	Reserved
12-8	NUM_FRAME_COUNT	Number of times to repeat NUM_R2_LENGTH_0, NUM_DUMMY_0, and ENUM_DATA_FORMAT_IN_0 (used for HSDPA). NUM_R2_LENGTH_1..5 are always 0 if this value is >1. If NUM_FRAME_COUNT > 1, this represents the number of frames in the packet, and all frames are the same length.
7	Reserved	Reserved
6-4	ENUM_DATA_FORMAT_OUT	Output data format. • 0 = Rel-99 DL DTX • 1 = 1-bit packed • 2 = 2-bit packed • 5 = TAC format (FDD only) • 7 = Unpacked – one data value per byte
3	FLAG_HALF_RATE	This flag tells the interleaver that it is pre-interleaving 1/2 rate convolution data. This flag is ignored if flagInOrder is 0.
2	FLAG_IN_ORDER	This flag tells the interleaver that it is interleaving data coming directly from the encoder instead of the rate-matcher sub-module. • 0 = input data is from rate-matcher • 1 = input data is from encoder
1-0	NUM_CONSTELLATION	This number selects constellation rearrangement format for 4/6 bit output symbol. It is the same as the “constellation version parameter b” in the 3GPP 25.212 spec. section 4.5.7 table 11 and table 11A. If no constellation re-arrangement needed, it should be set to 0. Values: 0, 1, 2, 3.

Figure 4-173. INT Local Packet Header Word 2

31 16	15 0
Reserved	NUM_DTX_VALUES

Table 4-149. INT Local Packet Header Word 2 Details

Bits	Field Name	Description
31-16	Reserved	Reserved
15-0	NUM_DTX_VALUES	Number of DTX to be appended at end of last frame. This value is specified in symbols. DTX is appended for last frame only.

Figure 4-174. INT Local Packet Header Word 3

31 27	26 24	23 21	20 16	15 0
Reserved	ENUM_DATA_FORMAT_IN_0	Reserved	NUM_DUMMY_0	NUM_R2_LENGTH_0

Table 4-150. INT Local Packet Header Word 3 Details

Bits	Field Name	Description
31-27	Reserved	Reserved
26-24	ENUM_DATA _FORMAT_IN_0	Input data format. • 0 = 2 bits per data value – BPSK with DTX or 4PAM • 1 = 1 bit per data value – BPSK • 2 = 2 bits per data value – QPSK • 3 = 3 bits per data value – 8-PSK • 4 = 4 bits per data value – 16-QAM • 6 = 6 bits per data value – 64-QAM
23-21	Reserved	Reserved
20-16	NUM_DUMMY_0	Number of dummy symbols per frame in the last Row. This is specified in units of symbols (not bits).
15-0	NUM_R2_LENGTH_0	Number of Rows in interleaver table per frame. This value is derived based on symbol (not bits). This value is inclusive of numDTXValue and numDummy. In case NUM_FRAME_COUNT is greater than 1 it implies that numR2Length[0] should be repeated for all frames within this packet.

Table 4-151. INT Local Packet Header Word 4 – Word 8

Word 4 – Word 8
The bit format for the param set 0 in INT local header word 3 is repeated for param sets 1 to 5, in local header words 4 to 8, respectively.

NOTE: The bit format for the param set 0 in INT local header word 3 is repeated for param sets 1 to 5, in local header words 4 to 8, respectively.

4.9.4 Error Bit Definitions

The INT error bits are defined in Table 4-152. These are the errors bits that will be recorded in the INT data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-152. INT Error Bit Definitions

BIT	NAME
6	Reserved
5	ERR_INT_FRAME_LENGTH. This error is set if one or more of following conditions are true. • Frame is smaller than numIntWays. • Incoming real bits are greater than or less than computed frame size.I • Frame length is greater than 37440 bits (not symbols) • Number of frames received from input is less or more than configured frames.
4	ERR_INT_CFG_CONSTELLATION. This error indicates that constellation rearrangement has been asked to be applied in non-supported modes as described in Table 4-146.
3	ERR_INT_CFG_DUMMY. This error is set in case numDummy[x] is greater or equal to number of interleaver table columns as defined by numIntWays.
2	ERR_INT_CFG_DTX. This error indicates the DTX has been asked to be inserted in non-DTX mode. This error is also set if number of DTX bits are more than actual real frame bits for last frame.
1	ERR_INT_CFG_INPUT_OUTPUT. This error is set if input to output data formatting is not matching as per Table 4-145 or Table 4-146 taking into account 1st and 2nd interleaver operation.
0	ERR_INT_CFG_ILLEGAL_RANGE. This error is set if one or more of following conditions are true. • Received configuration value(s) in one or more parameter(s) are outside of legal range. • Frame count is zero. • Frame count is greater than 1 for 1st interleaver. • Valid R2[x] are zero's. • flagInOrder is set for 2nd Interleaver.

4.10 DNT

The DNT submodule performs the following tasks:

- 2nd De-interleaving operation for WCDMA and TD-SCDMA
- Physical channel segmentation
- Removing DTX bits for Rel-99
- Constellation rearrangement for 2nd De-interleaving
- Descramble operation.

4.10.1 Data Format

4.10.1.1 Input Data

Input Data is packed into 128-bit values with each value representing a constellation point which may have 8,16 or 32 bits. The first constellation point is in the least significant bits of the 128-bit word. When NumFrameCount>1 each input Frame will be separately aligned to start on a 128-bit boundary, additional filler values are to be ignored. This padding allows the deInterleaver to also perform Physical Channel DeMapping on 128-bit aligned Frames.

4.10.1.2 Output Data

Output data consists of bytes packed into 128-bit words with the first byte in the least significant position. Data is packed continuously with each frame even for NumFrameCount>1 packed without padding.

4.10.2 Memory Mapped Registers

The only memory mapped registers are the data logger registers and are described in the data logger section.

4.10.3 Packet Header Configuration Fields

The packet header configuration fields for the DNT submodule are described in the following figures and table s.

Figure 4-175. DNT Local Packet Header Word 0

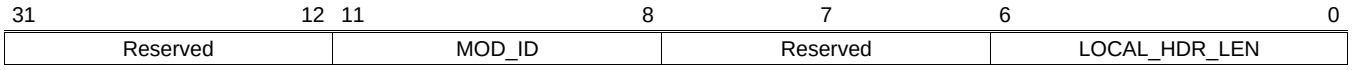


Table 4-153. DNT Local Packet Header Word 0 Details

Bits	Field Name	Description
31-12	Reserved	Reserved
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7	Reserved	Reserved
6-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

Figure 4-176. DNT Local Packet Header Word 1

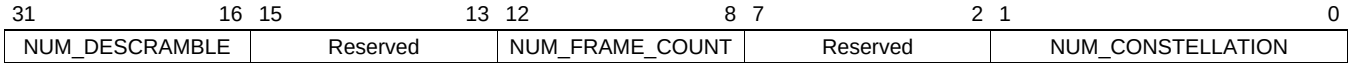


Table 4-154. DNT Local Packet Header Word 1 Details

Bits	Field Name	Description
31-16	NUM_DESCRAMBLE	Descrambler generator initialization value. • 0 = do not descramble • 1 = descramble.
15-13	Reserved	Reserved
12-8	NUM_FRAME_COUNT	Number of times to repeat NUM_R2_LENGTH_0, NUM_DUMMY_0, and ENUM_DATA_FORMAT_0 (used for HSDPA). NUM_R2_LENGTH_1..5 are always 0 if this value is >1. If NUM_FRAME_COUNT > 1, this represents the number of frames in the packet, and all frames are the same length.
7-2	Reserved	Reserved
1-0	NUM_CONSTELLATION	This number selects constellation rearrangement format for 16QAM. Values: 0, 1, 2, 3.

Figure 4-177. DNT Local Packet Header Word 2

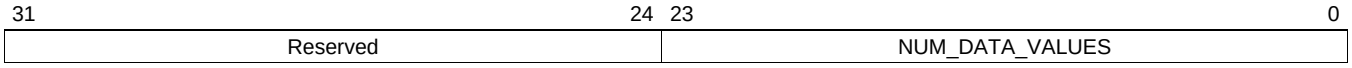


Table 4-155. DNT Local Packet Header Word 2 Details

Bits	Field Name	Description
31-24	Reserved	Reserved
23-0	NUM_DATA_VALUES	This count indicates total number of soft bits (bytes) for given packet. This must match with total number of frames multiplied by number of soft bits per frame. Module can accept 2^24 -1 soft bits.

Figure 4-178. DNT Local Packet Header Word 3

31 30	29 24	23 21	20 16	15 0
Reserved	ENUM_DATA_FORMAT_0	Reserved	NUM_DUMMY_0	NUM_R2_LENGTH_0

Table 4-156. DNT Local Packet Header Word 3 Details

Bits	Field Name	Description
31:30	Reserved	Reserved
29-24	ENUM_DATA _FORMAT_0	Data format. • 0 = QPSK • 8 = BPSK • 16 = 4PAM • 32 = 16-QAM
23-21	Reserved	Reserved
20-16	NUM_DUMMY_0	Number of dummy symbols per frame in the last Row. This is specified in units of symbols (not bits).
15-0	NUM_R2_LENGTH_0	Number of Rows in de-interleaver table per frame. This value is derived based on symbol (not bits). This value is inclusive of numDTXValue and numDummy.

Table 4-157. DNT Local Packet Header Word 4 – Word 8

Word 4 – Word 8
The bit format for the param set 0 in DNT local header word 3 is repeated for param sets 1 to 5, in local header words 4 to 8, respectively.

NOTE: The bit format for the param set 0 in INT local header word 3 is repeated for param sets 1 to 5, in local header words 4 to 8, respectively.

4.10.4 Error Bit Definitions

The DNT error bits are defined in Table 4-158. These are the errors bits that will be recorded in the DNT data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-158. DNT Error Bit Definitions

BIT	NAME
6	Reserved
5	ERR_INT_FRAME_LENGTH. This error is set if one or more of following conditions are true. • Number of ingress bits per packet is not equal to frame size * frame length. • numDataValue do not match with frame size * frame length • Frame length is greater than 39936 bits (not symbols)
4	ERR_INT_CFG_CONSTELLATION. This error indicates that constellation rearrangement has been asked to be applied in non-supported modes as described in Table 4-146.
3	ERR_INT_CFG_DUMMY. This error is set in case numDummy[x] is greater or equal to number of interleaver table columns as defined by numIntWays.
2	ERR_INT_CFG_DTX. This error indicates the DTX has been asked to be inserted in non-DTX mode. This error is also set if number of DTX bits are more than actual real frame bits for last frame.
1	ERR_INT_CFG_INPUT_OUTPUT. This error is set if input to output data formatting is not matching as per Table 4-145 or Table 4-146 taking into account 1st and 2nd interleaver operation.
0	ERR_INT_CFG_ILLEGAL_RANGE. This error is set if one or more of following conditions are true. • Received configuration value(s) in one or more parameter(s) are outside of legal range. • Frame count is zero. • Frame count is greater than 1 for 1st interleaver. • Valid R2[x] are zero's. • flagInOrder is set for 2nd Interleaver.

4.11 COR

The COR submodule performs correlation to either accelerate PUCCH decoding for LTE (for formats 2, 2a, or 2b) or perform final despreading for WCDMA / TD-SCDMA. In Keystone-II devices, the COR sub-module can also perform correlation for LTE PUCCH format 3.

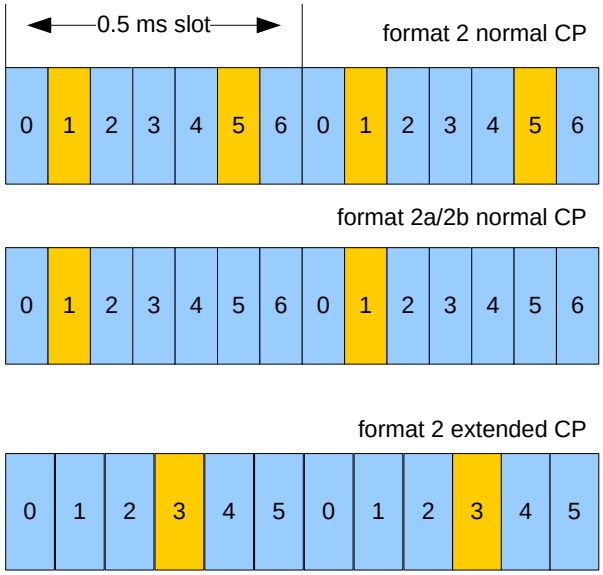
4.11.1 Data Format

4.11.1.1 PUCCH Correlation

This feature correlates complex received sequences with a set of QPSK sequences. The BCP returns the four largest magnitude squared values, their corresponding indices which would correspond to the most likely transmitted sequence, and the mean magnitude squared value.

For LTE PUCCH format 2/2a/2b, each user transmits a sequence of up to 13 bits on up to 10 data symbols and up to 4 pilot symbols as shown in Figure 4-179. The blue symbols are data and the yellow symbols are pilots.

Figure 4-179. PUCCH 2/2a/2b Formats



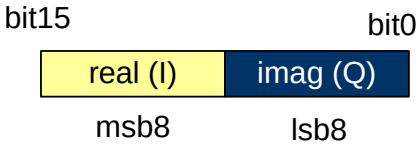
In the case of format 2a/2b, the data will be sent to the BCP two (format 2a) or four (format 2b) times for each of the possible ACK/NACK bit possibilities. In this case, symbol 5 is treated as one of the pilots. In the case of the extended cyclic prefix, two of the four pilots are set to 0.

The COR submodule can be configured to use a “trick” taking advantage of the structure in the default Reed-Muller matrix columns M(i,0) and M(i,5). Column M(i,0) is all ones meaning that a change of the original input bit a0 results in all 20 coded bits flipping. This in turn results in the QPSK-modulated signal being multiplied by –1. Similarly, column M(i,5) is split into half 0s and half 1s, meaning that the first 10 output bits do not depend on this input bit at all. The second 10 output bits are affected as previously mentioned by input bits 0 and 5. Effectively, this means we only need to search 2^(A- 2) sequences instead of 2^A when A ≤ 5 and by 2^(A-1) sequences when A < 5.

4.11.1.2 PUCCH - Input Data

The input data always has 14 × Nr (number of receive antennas) complex samples. Each value is 16 bits (8-bit I and 8-bit Q) in the format shown in Figure 4-180.

Figure 4-180. PUCCH Input Data Value Format



The PUCCH input data consists of pilot symbols (p) and data symbols (y) as described in Table 4-159.

Table 4-159. PUCCH Input Data Value Types

Parameter	#bits	Description
p	Nr × 4 × 8I/8Q	4 input pilots, in 2s compliment format (from +127 to -128)
y	Nr ×10 × 8I/8Q	10 input data points in 2s compliment format (from +127 to -128)

In LTE PUCCH, the pilot symbols are always the same transmitted symbol and they are 45 degrees rotated from the data symbols. To use the BCP, the DSP must undo the rotation before sending the input to the BCP (multiply by (1 + j)/sqrt(2)). Correspondingly, the COR Control Register described in Section 4.11.4.14 should be set to 0x1B.

The packing of the input data is shown in Table 4-160, where $p_{i,j,n}$ and $y_{i,j,n}$ is for slot *i*, symbol *j*, and antenna *n* (up to 8 antennas). Qw means “quad word” and is 128-bits, or 16 bytes. Fields marked as “RES” are reserved and are not used. They should be written as 0x0. The COR submodule always expects 10 data symbols and 4 pilots. If there are fewer pilots or data subcarriers, the inputs to the correlation submodule for unused symbols/pilots must be set to 0.

Table 4-160. PUCCH Input Data Packing Format

Bits	0-15	16-31	32-47	48-63	64-79	80-95	96-11	112-127
Qw0	$y_{0,0,0}$	$y_{0,1,0}$	$y_{0,2,0}$	$y_{0,3,0}$	$y_{0,4,0}$	$y_{1,0,0}$	$y_{1,1,0}$	$y_{1,2,0}$
Qw1	$y_{1,3,0}$	$y_{1,4,0}$	Reserved	Reserved	$p_{0,0,0}$	$p_{0,1,0}$	$p_{1,0,0}$	$p_{1,1,0}$
Qw2	$y_{0,0,1}$	$y_{0,1,1}$	$y_{0,2,1}$	$y_{0,3,1}$	$y_{0,4,1}$	$y_{1,0,1}$	$y_{1,1,1}$	$y_{1,2,1}$
Qw3	$y_{1,3,1}$	$y_{1,4,1}$	Reserved	Reserved	$p_{0,0,1}$	$p_{0,1,1}$	$p_{1,0,1}$	$p_{1,1,1}$
Qw4	$y_{0,0,2}$	$y_{0,1,2}$	$y_{0,2,2}$	$y_{0,3,2}$	$y_{0,4,2}$	$y_{1,0,2}$	$y_{1,1,2}$	$y_{1,2,2}$
Qw5	$y_{1,3,2}$	$y_{1,4,2}$	Reserved	Reserved	$p_{0,0,2}$	$p_{0,1,2}$	$p_{1,0,2}$	$p_{1,1,2}$
Qw6	$y_{0,0,3}$	$y_{0,1,3}$	$y_{0,2,3}$	$y_{0,3,3}$	$y_{0,4,3}$	$y_{1,0,3}$	$y_{1,1,3}$	$y_{1,2,3}$
Qw7	$y_{1,3,3}$	$y_{1,4,3}$	Reserved	Reserved	$p_{0,0,3}$	$p_{0,1,3}$	$p_{1,0,3}$	$p_{1,1,3}$
Qw8	$y_{0,0,4}$	$y_{0,1,4}$	$y_{0,2,4}$	$y_{0,3,4}$	$y_{0,4,4}$	$y_{1,0,4}$	$y_{1,1,4}$	$y_{1,2,4}$
Qw9	$y_{1,3,4}$	$y_{1,4,4}$	Reserved	Reserved	$p_{0,0,4}$	$p_{0,1,4}$	$p_{1,0,4}$	$p_{1,1,4}$
Qw10	$y_{0,0,5}$	$y_{0,1,5}$	$y_{0,2,5}$	$y_{0,3,5}$	$y_{0,4,5}$	$y_{1,0,5}$	$y_{1,1,5}$	$y_{1,2,5}$
Qw11	$y_{1,3,5}$	$y_{1,4,5}$	Reserved	Reserved	$p_{0,0,5}$	$p_{0,1,5}$	$p_{1,0,5}$	$p_{1,1,5}$
Qw12	$y_{0,0,6}$	$y_{0,1,6}$	$y_{0,2,6}$	$y_{0,3,6}$	$y_{0,4,6}$	$y_{1,0,6}$	$y_{1,1,6}$	$y_{1,2,6}$
Qw13	$y_{1,3,6}$	$y_{1,4,6}$	Reserved	Reserved	$p_{0,0,6}$	$p_{0,1,6}$	$p_{1,0,6}$	$p_{1,1,6}$
Qw14	$y_{0,0,7}$	$y_{0,1,7}$	$y_{0,2,7}$	$y_{0,3,7}$	$y_{0,4,7}$	$y_{1,0,7}$	$y_{1,1,7}$	$y_{1,2,7}$
Qw15	$y_{1,3,7}$	$y_{1,4,7}$	Reserved	Reserved	$p_{0,0,7}$	$p_{0,1,7}$	$p_{1,0,7}$	$p_{1,1,7}$

4.11.1.3 PUCCH - Output Data

The PUCCH output data consists of the data value types described in Table 4-161.

Table 4-161. PUCCH Output Data Value Types

Parameter	#bits	Description
a0	13	highest correlation's input bit sequence a(A-1),...,a1,a0 in the A least-significant bits of the allocated 16 bits
a1	13	next highest correlation input sequence
a2	13	third highest correlation input sequence
a3	13	fourth highest correlation input sequence
corr0	25	correlation (magnitude-squared) corresponding to the highest correlation input sequence
corr1	25	correlation (magnitude-squared) corresponding to the next highest correlation input sequence
corr2	25	correlation (magnitude-squared) corresponding to the third highest correlation input sequence
corr3	25	correlation (magnitude-squared) corresponding to the fourth highest correlation input sequence
mean	32	mean of all correlations

The packing of the output data is shown in Table 4-162. Most of the parameters are smaller than their allocated space, so the unused most-significant bits of each field are reserved and return 0s.

Table 4-162. PUCCH Output Data Packing Format

Bits	0-15	16-31	32-47	48-63	64-79	80-95	96-11	112-127
Qw0	a_0	a_1	a_2	a_3	$corr_0$		$corr_1$	
Qw1	$corr_2$		$corr_3$		mean		Reserved	

4.11.1.4 Final Despreading

In some WCDMA and TD-SCDMA modes the actual spreading factor is not known during the receive process and it is necessary to receive and partially de-spread the data at the lowest spreading factor, and then perform the final portion of the de -spreading later to reduce the data to the final rate. The COR submodule can perform this final de -spreading. The input is a number of real or complex symbols along with the spreading factor (SF) ratio. The number of output symbols is then the number of input symbols divided by the SF ratio, where each group of SF_ratio symbols have been added together to form an output symbol.

4.11.1.4.1 Input Data

An incoming packet contains all the physical channels associated with 2 ms, 5 ms, or 10 ms of data for one CCTrCh Rel-99 channel or one HSUPA transport channel as appropriate. The de-spreader handles up to 6 physical channels in a single packet, each with its own spreading factor ratio and length. For WCDMA, the input is always 16-bit real values. For TD-SCDMA, the inputs are 32-bit complex values (16-bit I and 16-bit Q). The order of I and Q within each symbol does not matter to the COR submodule de-spreader function since it simply combines (adds) the symbols. (The I/Q ordering can be configured in the soft slicer (SSL) submodule which will use output data from the de-spreader.)

Input Data is packed into 128-bit words with the first received value in the least significant bits. Each physical channel must start on a 128-bit boundary. In order to pass multiple physical channels as a two dimensional block, with each physical channel separately aligned on a 128-bit boundary, padding may be required at the end of each physical channel. These additional pad values are excluded in the input length parameter. The following is an example of how two input physical channels containing 10 and 12 real samples would be packed together. The input symbols are labelled S0 for the first physical channel and S1 for the second physical channel.

Table 4-163. Example COR De-Spreading Input (real inputs)

Bits	127 - 112	111 - 96	95 - 80	79 - 64	63 - 48	47 - 32	31 - 16	15 - 0
Word0	$S0_7$	$S0_6$	$S0_5$	$S0_4$	$S0_3$	$S0_2$	$S0_1$	$S0_0$
Word1	Pad Bits						$S0_9$	$S0_8$
Word2	$S1_7$	$S1_6$	$S1_5$	$S1_4$	$S1_3$	$S1_2$	$S1_1$	$S1_0$
Word3					$S1_{11}$	$S1_{10}$	$S1_9$	$S1_8$

4.11.1.4.2 Output Data

Output data is packed into 128-bit words, with the first output value in the least significant bits. Data is always aligned to start on a 128-bit boundary. Data values are either 16-bit real soft symbols or 32-bit complex (16-bit I and 16-bit Q) format, depending on the input format. In order to output multiple physical channels as a two dimensional block, with each physical channel separately aligned on a 128-bit boundary, padding may be added at the end of each physical channel.

The following is an example of how two output physical channels containing 5 and 3 real output symbols would be packed together. The output symbols are labelled SO0 for the first physical channel and SO1 for the second physical channel.

Table 4-164. Example COR De-Spreading Output (real inputs)

Bits	127 - 112	111 - 96	95 - 80	79 - 64	63 - 48	47 - 32	31 - 16	15 - 0
Word	Padding Bits			$SO0_4$	$SO0_3$	$SO0_2$	$SO0_1$	$SO0_0$
Word						$SO1_2$	$SO1_1$	$SO1_0$

4.11.2 Memory Mapped Registers

The memory mapped registers inside the COR submodule are described in this section. There are also the standard data logger registers (not shown in this section) that are described in the data logger section.

4.11.3 Memory Map

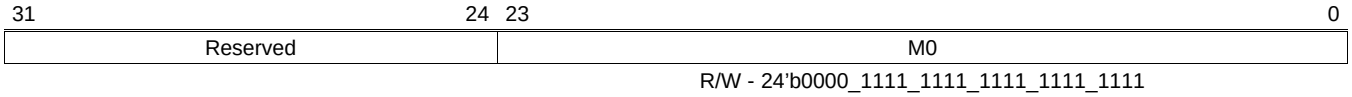
Table 4-165. COR Memory Map

ADDRESS OFFSET	REGISTER NAME	RESET
0x0C00	M0	0x000FFFFFu
0x0C04	M1	0x0005A933u
0x0C08	M2	0x00010E5Au
0x0C0C	M3	0x0006339Cu
0x0C10	M4	0x0007C3E0u
0x0C14	M5	0x000FFC00u
0x0C18	M6	0x000D8E64u
0x0C1C	M7	0x0004F5B0u
0x0C20	M8	0x000218ECu
0x0C24	M9	0x0001B746u
0x0C28	M10	0x0000FFFFu
0x0C2C	M11	0x00033FFFu
0x0C30	M12	0x0003FFFCu
0x0C34	Control	0x00000000u

4.11.4 Register Descriptions

4.11.4.1 M0 Register

Figure 4-181. M0 Register



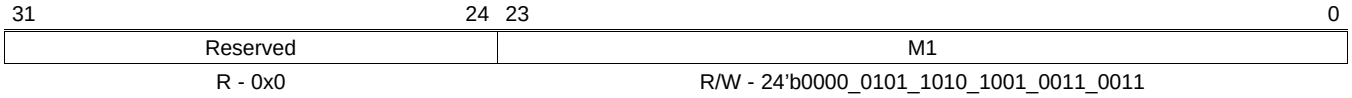
Legend: R = Read only; W = Write only

Table 4-166. M0 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M0	Register M0 holds the 0 column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M0 equal to [b23~b0] of the 0 column in the (32,O) code • The second time, set Bits [23:0] of register M0 equal to [b15~b0, b31~b24] of the 0 column in the (32,O) code

4.11.4.2 M1 Register

Figure 4-182. M1 Register



Legend: R = Read only; W = Write only

Table 4-167. M1 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M1	Register M1 holds the 1st column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M1 equal to [b23~b0] of the 1st column in the (32,O) code • The second time, set Bits [23:0] of register M1 equal to [b15~b0, b31~b24] of the 1st column in the (32,O) code

4.11.4.3 M2 Register

Figure 4-183. M2 Register

31 24	23 0
Reserved	M2
R - 0x0	R/W - 24'b0000_0001_0000_1110_0101_1010

Legend: R = Read only; W = Write only

Table 4-168. M2 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M2	Register M2 holds the 2nd column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M2 equal to [b23~b0] of the 2nd column in the (32,O) code • The second time, set Bits [23:0] of register M2 equal to [b15~b0, b31~b24] of the 2nd column in the (32,O) code

4.11.4.4 M3 Register

Figure 4-184. M3 Register

31 24	23 0
Reserved	M3
R - 0x0	R/W - 24'b0000_0110_0011_0011_1001_1100

Legend: R = Read only; W = Write only

Table 4-169. M3 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M3	Register M3 holds the 3rd column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M3 equal to [b23~b0] of the 3rd column in the (32,O) code • The second time, set Bits [23:0] of register M3 equal to [b15~b0, b31~b24] of the 3rd column in the (32,O) code

4.11.4.5 M4 Register

Figure 4-185. M4 Register

31 24	23 0
Reserved	M4
R - 0x0	R/W - 24'b0000_0111_1100_0011_1110_0000

Legend: R = Read only; W = Write only

Table 4-170. M4 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M4	Register M4 holds the 4th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M4 equal to [b23~b0] of the 4th column in the (32,O) code • The second time, set Bits [23:0] of register M4 equal to [b15~b0, b31~b24] of the 4th column in the (32,O) code

4.11.4.6 M5 Register

Figure 4-186. M5 Register

31 24	23 0
Reserved	M5
R - 0x0	R/W - 24'b0000_1111_1111_1100_0000_0000

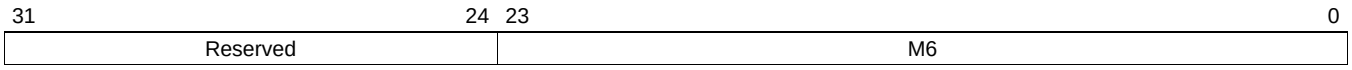
Legend: R = Read only; W = Write only

Table 4-171. M5 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M5	Register M5 holds the 5th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M5 equal to [b23~b0] of the 5th column in the (32,O) code • The second time, set Bits [23:0] of register M5 equal to [b15~b0, b31~b24] of the 5th column in the (32,O) code

4.11.4.7 M6 Register

Figure 4-187. M6 Register



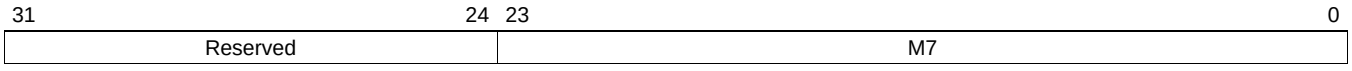
Legend: R = Read only; W = Write only

Table 4-172. M6 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M6	Register M6 holds the 6th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M6 equal to [b23~b0] of the 6th column in the (32,O) code • The second time, set Bits [23:0] of register M6 equal to [b15~b0, b31~b24] of the 6th column in the (32,O) code

4.11.4.8 M7 Register

Figure 4-188. M7 Register



Legend: R = Read only; W = Write only

Table 4-173. M7 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M7	Register M7 holds the 7th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M7 equal to [b23~b0] of the 7th column in the (32,O) code • The second time, set Bits [23:0] of register M7 equal to [b15~b0, b31~b24] of the 7th column in the (32,O) code

4.11.4.9 M8 Register

Figure 4-189. M8 Register

31	24	23	0
Reserved		M8	
R - 0x0		R/W - 24'b0000_0010_0001_1000_1110_1100	

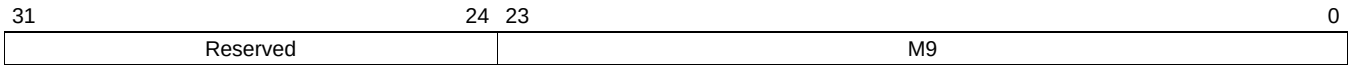
Legend: R = Read only; W = Write only

Table 4-174. M8 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M8	Register M8 holds the 8th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M8 equal to [b23~b0] of the 8th column in the (32,O) code • The second time, set Bits [23:0] of register M8 equal to [b15~b0, b31~b24] of the 8th column in the (32,O) code

4.11.4.10 M9 Register

Figure 4-190. M9 Register



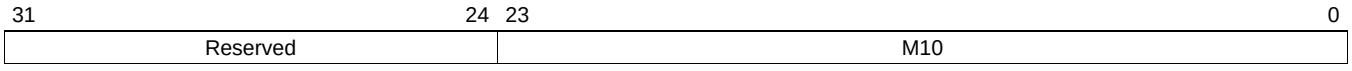
Legend: R = Read only; W = Write only

Table 4-175. M9 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M9	Register M9 holds the 9th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M9 equal to [b23~b0] of the 9th column in the (32,O) code • The second time, set Bits [23:0] of register M9 equal to [b15~b0, b31~b24] of the 9th column in the (32,O) code

4.11.4.11 M10 Register

Figure 4-191. M10 Register



Legend: R = Read only; W = Write only

Table 4-176. M10 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M10	Register M10 holds the 10th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set two times during PUCCH format 3 decoding: • The first time, set Bits [23:0] of register M10 equal to [b23~b0] of the 10th column in the (32,O) code • The second time, set Bits [23:0] of register M10 equal to [b15~b0, b31~b24] of the 10th column in the (32,O) code

4.11.4.12 M11 Register

Figure 4-192. M11 Register

31	24	23	0
Reserved		M11	
R - 0x0		R/W - 24'b0000_0011_0011_1111_1111_1111	

Legend: R = Read only; W = Write only

Table 4-177. M11 Register Details

Bits	Name	DESCRIPTION
31-24	Reserved	Read as 0
23-0	M11	Register M11 holds the 11th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set to zero.

4.11.4.13 M12 Register

Figure 4-193. M12 Register

31 24	23 0
Reserved	M12
R - 0x0	R/W -24'b0000_0011_1111_1111_1111_1100

Legend: R = Read only; W = Write only

Table 4-178. M12 Register Details

Bits	Name	Description
31-24	Reserved	Read as 0
23-0	M12	Register M12 holds the 12th column of the Reed-Muller Table. The reset value is the default for standard LTE PUCCH format 2. When PUCCH_MODE = 0 or 1 (PUCCH format 2): • Bits [23:20]: Reserved • Bits [19:0]: – MSB = M(19,column) – LSB = M(0,column) When PUCCH_MODE = 2 (PUCCH format 3), this register should be set to zero.

4.11.4.14 Control Register

Figure 4-194. Control Register

31 8	7 6	5 4	3 2	1 0
Reserved	QPSK Map 11	QPSK Map 10	QPSK Map 01	QPSK Map 00
R - 0x0	R/W - 0x0	R/W - 0x0	R/W - 0x0	R/W - 0x0

Legend: R = Read only; W = Write only

Table 4-179. Control Register Details

Bits	Name	Description
31-8	Reserved	Read as 0
7-6	QPSK Map 11	This field controls how the 2-bit pair in the scrambled sequence maps to QPSK symbols, when the 2-bit pair is $b_{2k} = 1$ and $b_{2k+1} = 1$ • 00b = +1 • 01b = +j • 10b = -j • 11b = -1
5-4	QPSK Map 10	This field controls how the 2-bit pair in the scrambled sequence maps to QPSK symbols, when the 2-bit pair is $b_{2k} = 1$ and $b_{2k+1} = 0$ • 00b = +1 • 01b = +j • 10b = -j • 11b = -1
3-2	QPSK Map 01	This field controls how the 2-bit pair in the scrambled sequence maps to QPSK symbols, when the 2-bit pair is $b_{2k} = 0$ and $b_{2k+1} = 1$ • 00b = +1 • 01b = +j • 10b = -j • 11b = -1

Table 4-179. Control Register Details (continued)

Bits	Name	Description
1-0	QPSK Map 00	This field controls how the 2-bit pair in the scrambled sequence maps to QPSK symbols, when the 2-bit pair is $b_{2k} = 0$ and $b_{2k+1} = 0$ • 00b = +1 • 01b = +j • 10b = -j • 11b = -1

4.11.5 Packet Header Configuration Fields

4.11.5.1 Word 0

Figure 4-195. Word 0

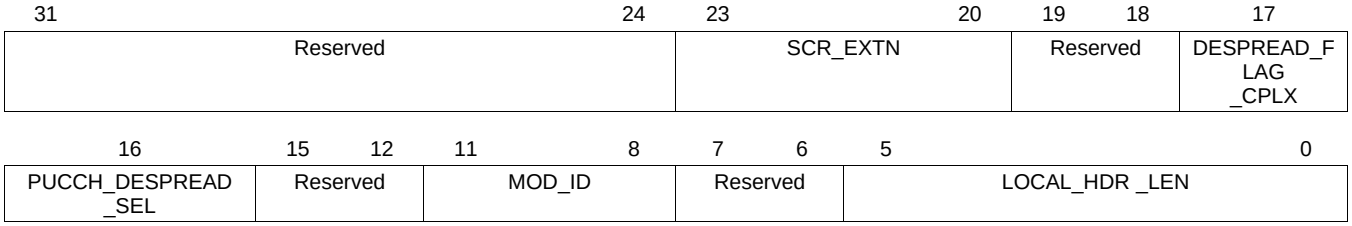


Table 4-180. Word 0 Details

Bits	Name	Description
31-24	Reserved	Write as 0.
23-20	SCR_EXTN	PUCCH_SCR_SEQ extension
19-18	Reserved	Reserved
17	DESPREAD_FLAG_CPLX	Complex indicator for de-spreading • 0 – Inputs are 16-bit real numbers • 1 – Inputs are 32-bit complex numbers
16	PUCCH_DESPREAD_SEL	Select between PUCCH and Final De-spreading functionality • 0 – PUCCH • 1 - De-spreading
15-12	Reserved	Reserved to get to 16 bit boundary.
11-8	MOD_ID	Module ID for this submodule (see Table 4-1 table)
7-6	Reserved	Reserved
5-0	LOCAL_HDR_LEN	Local header length in 32-bit words, excluding this word (Word 0).

4.11.5.2 Word 1

Figure 4-196. Word 1

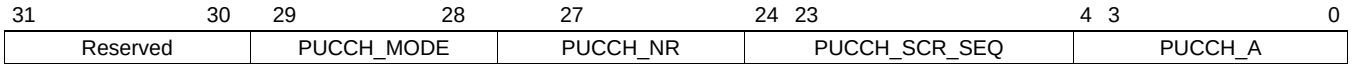


Table 4-181. Word 1 Details

Bits	Name	Description
31-30	Reserved	Write as 0
29-28	PUCCH_MODE	Operational mode. • 0 = PUCCH format 2, magnitude squared with PUCCH “trick”: return 4 largest magnitude squared values, the corresponding indices, and the overall mean. • 1 = PUCCH format 2, magnitude squared but PUCCH “trick” disabled: return 4 largest magnitude squared values, the corresponding indices, and the overall mean. • 2 = PUCCH format 3: return all magnitude squared values. In this mode, the COR only calculates all the possible magnitude squared values but does not compare them to find the maximum. The number of returned values is 2^PUCCH_A.
27-24	PUCCH_NR	Number of receive antennas (1 to 8)

Table 4-181. Word 1 Details (continued)

Bits	Name	Description
23-4	PUCCH_SCR_SEQ	Scrambling Code. The 20-bit scrambling sequence with PUCCH_SCR_SEQ[0] in the least significant bit of the field, i.e. bit 4 in the word, PUCCH_SCR_SEQ[19] in the most significant bit of the field, i.e. bit 23 in the word. (set to all 0s to disable). • PUCCH_SCR_SEQ[0] is XORd with b0 • PUCCH_SCR_SEQ[1] is XORd with b1 • …. • PUCCH_SCR_SEQ[19] is XORd with b19
3-0	PUCCH_A	Number of user bits. The number of information bits (1 to 13) – the number of codes over which to search.

4.11.5.3 Words 2-7

Figure 4-197. Words 2-7

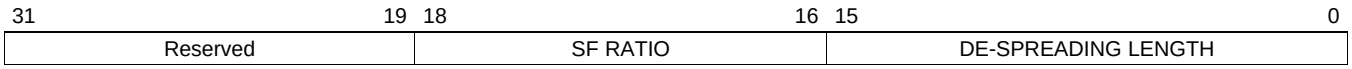


Table 4-182. Words 2-7 Details

Bits	Name	Description
31-19	Reserved	Write as 0.
18-16	SF Ratio	The de-spreading ratio. The ratio is 2SF ratio or as listed below. Value – Ratio: • 0 = 1 • 1 – 2 • 2 – 4 • 3 – 8 • 4 – 16 • 5 – 32 • 6 – 64 • 7 - 128
15-0	De-spreading Length	The number of samples given for de-spreading. Largest possible is $2^{16} - 1$ = 65,53 5 samples

NOTE: Note that words 2-7 are identical, except that word 2 has the parameters for the first physical channel, word 3 has the parameters for the 2nd physical channel, etc. All blocks must have either real or complex inputs as indicated by the DESPREAD_FLAG_CPLX flag in config word 0.

4.11.6 Error Bit Definitions

The COR error bits are defined in Table 4-183. These are the errors bits that will be recorded in the COR data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-183. COR Error Bit Definitions

BIT	NAME
6	A de-spreading packet is received where the length of all de-spreading channels is 0.
5	An incorrect number of input samples were received for a de-spreading packet. This means that the number of input samples didn't equal the number expected due to the length of all physical channels, or that the number of samples for one of the channels wasn't a multiple of the spreading factor.
4	The number of pilot/data symbols received does not equal what was expected based on PUCCH_NR.
3	Invalid packet header configuration. PUCCH_NR is 0.
2	Invalid packet header configuration. PUCCH_NR is greater than 8.
1	Invalid packet header configuration. PUCCH_A is 0.
0	Invalid packet header configuration. PUCCH_A is greater than 13.

4.12 DIO

Normally, packets enter and exit the BCP through the Packet DMA and when they do, the location and length of the data is determined by pointers in the Navigator descriptors. However, to support features like LLR combining and transport channel multiplexing, a direct I/O override (DIO) is provided. The override allows the DSP to directly specify a set of addresses and lengths rather than a Navigator descriptor and queue. There are two instances of the DIO interface in the BCP – one embedded inside the rate de-matching (RD) submodule and another that is a stand-alone submodule in the BCP.

The DIO DMA inside RD is not directly visible to the software user. If HARQ LLR reads or writes are enabled, RD will use the DIO interface to fetch or write LLRs. The address and length of the LLRs for reading and writing are part of the RD header. In this case, there is only one read and one write address for every BCP packet. It is assumed that LLRs are stored contiguously in one data buffer at any location that the BCP can read or write including local or shared L2 memory and DDR. LLRs are always bytes so there is no format conversion in the RD DIO except to handle endian conversion.

In the case of the standalone DIO submodule, it has a local header that tells it whether to read or write data and includes the length(s) and address(es). When used, the DIO submodule replaces the Packet DMA for the reading or writing of payload data. When the DIO submodule is instructed to read data, the BCP packet must pass through the DIO submodule before it reaches the submodule for which the DIO data is used. Data read by the DIO submodule is appended to the end of the packet as payload – DIO assumes that there was no packet payload prior to arriving at the DIO submodule (if there was, it is dropped and overwritten). This mode of using DIO can be used with any of the BCP submodules but is intended to read the transport channel data for the second stage of R99 downlink processing (radio frame processing).

The DIO submodule can read from up to 6 different addresses, each with its own length specified in bytes. The addresses must be on a 16 byte boundary, i.e., the four least-significant bits of the addresses must be 0. If there is more than one address, the DIO will concatenate and pack the reads onto the 128-bit BCP bus. DIO can read any number of bytes. The first byte that DIO reads will always be in position 0 on the 128-bit BCP bus.

When the DIO submodule is instructed to write data, it will generally be the last submodule visited before the traffic manager. Like all other submodules, DIO strips off its local header. The DIO submodule then writes the payload in the packet to a single address indicated in the header – DIO write operation only support s a single destination address. The address must be on a 16 byte boundary, i.e. the four least-significant bits of the address must be 0. DIO then strips off the payload from the packet before forwarding the packet. The DIO will write the packet payload out as one block of data. It will send as much data as is in the payload received and will not specify or rely on a write count. DIO_ADDR_0 is the starting address of the write block. The packet continues to travel after the DIO has stripped its payload. This way, the DSP can be notified that the packet is complete because it will be written to a queue like all other Navigator packets.

Format and endian conversion is handled in DIO in a similar fashion as in the Packet DMA. The packet header determines the format and the endian control. Endianness is determined by the value of an input pin on the device.

4.12.1 Memory Mapped Registers

There is only one memory mapped register other than the standard data logger registers in the DIO submodule. It is located at address BCP base address + 0x0400.

Figure 4-198. DIO VBUSM PRIORITY Register

31	3	2	0
Reserved		PRIORITY	
R		R/W - 0	

Legend: R = Read only; W = Write only

Table 4-184. DIO VBUSM PRIORITY Register Details

Bits	Name	Description
31-3	Reserved	Reserved
2-0	Priority	0 = highest, 7 = lowest reset value = 0 (highest)

4.12.2 Packet Header Configuration Fields

The DIO submodule uses a variable length local header depending on the number of blocks to be transferred. The following figures and tables show the fields needed for the DIO. There is one control word plus 2 words per DMA block transfer needed. The maximum number of DMA blocks is 6.

Figure 4-199. DIO Packet Header Configuration Fields - Word 0

31 28	27 25	24	23 21	20 19	18 16	15 0
Reserved	DIO_BLK_CNT	DIO_RD_WR	DIO_FORMAT	DIO_ENDIAN	Reserved	LOCAL_HDR

Table 4-185. DIO Packet Header Configuration Fields - Word 0 Details

Bits	Name	Description
31-28	Reserved	Reserved
27-25	DIO_BLK_CNT	Number of DMA blocks for this operation
24	DIO_RD_WR	Read write control. 0 = write, 1 = read.
23-21	DIO_FORMAT	Format swap control • 0 = no swap • 1 = 32 bit reversal • 2 = 16 bit reversal • 3 = 8 bit reversal • 4 = no swap • 5 = 32 bit word swap • 6 = 16 bit word swap • 7 = 8 bit byte swap
20-19	DIO_ENDIAN	Endian swap control • 0 = no swap • 1 = 32 bit byte swap • 2 = 16 bit byte swap • 3 = 8 bit byte swap
18-16	Reserved	Reserved
15-0	LOCAL_HDR	Standard local header.

Figure 4-200. DIO Packet Header Configuration Fields - Word 1

31 0
DIO_ADDR_0

Table 4-186. DIO Packet Header Configuration Fields - Word 1 Details

Bits	Name	Description
31-0	DIO_ ADDR_0	Address for the first DMA block 0

Figure 4-201. DIO Packet Header Configuration Fields - Word 2

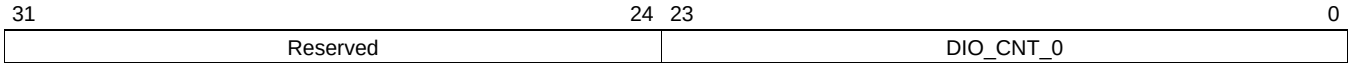


Table 4-187. DIO Packet Header Configuration Fields - Word 2 Details

Bits	Name	Description
31-24	Reserved	Reserved
23-0	DIO_ CNT_0	Number of bytes needed to be accessed in a read operation. Multiple of 16 bytes allowed.

Figure 4-202. DIO Packet Header Configuration Fields - Word 3

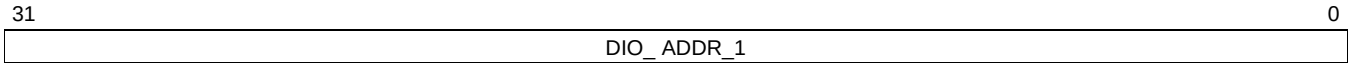


Table 4-188. DIO Packet Header Configuration Fields - Word 3 Details

Bits	Field	Description
31-0	DIO_ ADDR_1	Address for the DMA block 1

Figure 4-203. DIO Packet Header Configuration Fields - Word 4

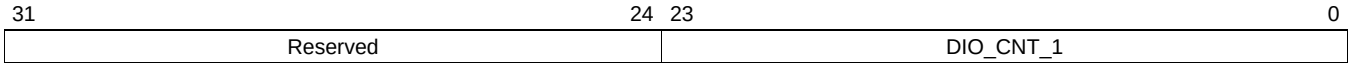


Table 4-189. DIO Packet Header Configuration Fields - Word 4 Details

Bits	Field	Description
31-24	Reserved	Reserved
23-0	DIO_ CNT_1	Number of bytes needed to be accessed in a read operation. Multiple of 16 bytes allowed.

Figure 4-204. DIO Packet Header Configuration Fields - Word 5

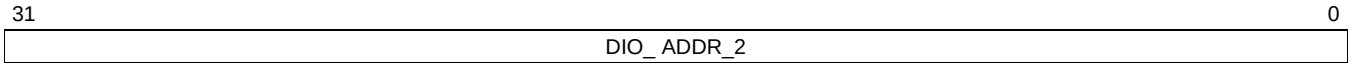


Table 4-190. DIO Packet Header Configuration Fields - Word 5 Details

Bits	Field	Description
31-0	DIO_ ADDR_2	Address for the DMA block 2

Figure 4-205. DIO Packet Header Configuration Fields - Word 6

31 24	23 0
Reserved	DIO_CNT_2

Table 4-191. DIO Packet Header Configuration Fields - Word 6 Details

Bits	Field	Description
31-24	Reserved	Reserved
23-0	DIO_ CNT_2	Number of bytes needed to be accessed in a read operation. Multiple of 16 bytes allowed.

Figure 4-206. DIO Packet Header Configuration Fields - Word 7

31 0
DIO_ ADDR_3

Table 4-192. DIO Packet Header Configuration Fields - Word 7 Details

Bits	Field	Description
31-0	DIO_ ADDR_3	Address for the DMA block 3

Figure 4-207. DIO Packet Header Configuration Fields - Word 8

31 24	23 0
Reserved	DIO_CNT_3

Table 4-193. DIO Packet Header Configuration Fields - Word 8 Details

Bits	Field	Description
31-24	Reserved	Reserved
23-0	DIO_ CNT_3	Number of bytes needed to be accessed in a read operation. Multiple of 16 bytes allowed.

Figure 4-208. DIO Packet Header Configuration Fields - Word 9

31 0
DIO_ ADDR_4

Table 4-194. DIO Packet Header Configuration Fields - Word 9 Details

Bits	Field	Description
31-0	DIO_ ADDR_4	Address for the DMA block 4

Figure 4-209. DIO Packet Header Configuration Fields - Word 10

31 24	23 0
Reserved	DIO_CNT_4

Table 4-195. DIO Packet Header Configuration Fields - Word 10 Details

Bits	Field	Description
31-24	Reserved	Reserved
23-0	DIO_ CNT_4	Number of bytes needed to be accessed in a read operation. Multiple of 16 bytes allowed.

Figure 4-210. DIO Packet Header Configuration Fields - Word 11

31	0
DIO_ ADDR_5	

Table 4-196. DIO Packet Header Configuration Fields - Word 11 Details

Bits	Field	Description
31-0	DIO_ ADDR_5	Address for the DMA block 5

Figure 4-211. DIO Packet Header Configuration Fields - Word 12

31 24	23 0
Reserved	DIO_CNT_5

Table 4-197. DIO Packet Header Configuration Fields - Word 12 Details

Bits	Field	Description
31-24	Reserved	Reserved
23-0	DIO_ CNT_5	Number of bytes needed to be accessed in a read operation. Multiple of 16 bytes allowed.

4.12.3 Error Bit Definitions

The DIO error bits are defined in Table 4-198. These are the errors bits that will be recorded in the DIO data logger RAM (if data logger is enabled) and generate an error interrupt (if enabled). Please see the debug usage section for more information.

Table 4-198. DIO Error Bit Definitions

BIT	NAME
6	Reserved
5	Reserved
4	Reserved
3	Reserved
2	Reserved
1	Reserved
0	Read configuration has a zero byte DIO_CNT_X in any of the read blocks or a DIO_BLK_CNT set to zero.

4.13 Packet DMA

4.13.1 Memory Mapped Registers

The memory mapped registers for the Packet DMA are described in the Navigator user's guide. The Packet DMA register region offsets for BCP are shown in Table 4-199. These offsets should be added to the BCP base address to obtain the full address.

Table 4-199. BCP Packet DMA Register Region Offsets

Region	Offset
Global Control	0x14000
Tx Channel Config	0x16000
Rx Channel Config	0x18000
Tx Scheduler Config	0x1A000
Rx Flow Config	0x1C000

Revision History

Changes from August 1, 2011 to May 30, 2015 **Page**

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265