WAVE521 HEVC and AVC Multi-encoder IP

Programmer's Guide

Version 1.4.0

WAVE521 HEVC and AVC Multi-encoder IP: Programmer's Guide

Version 1.4.0

Revision History

Date	Revision	Change
2018/7/12	0.9.0	Draft version generated
2018/7/31	1.0.0	Release version generated
2018/8/31	1.1.0	S2FME_OFF of CMD_ENC_BG_PARAM[29] was defined. Common I/O registers were set aside from each command I/O section.
2018/10/22	1.2.0	The latest host interface registers were applied to this document.
2018/10/31	1.3.0	CMD_CREATE_INST_SUB_FRM_SYNC and ENC_PIC_SUB_FRAME_SYNC_IF registers were added to support subframe synchronization.
2018/11/26	1.4.0	Setting AXI ID in VPU Initialization (optional) was described

Proprietary Notice

Address and Phone Number

Chips&Media
V&S Tower, 11/12/13th FL, 891-46, Daechi-dong, Gangnam-gu, 135-280
Seoul, Korea
Tel: +82-2-568-3767
Fax: +82-2-568-3767

Homepage: *http://www.chipsnmedia.com*

Table of Contents

List of Figures

List of Tables

List of Examples

Preface

This preface introduces the WAVE521 Programmer's Guide and its reference documentation. It contains the following sections:

1. About This Document

This document is the programmer's guide for WAVE521 HEVC and AVC Encoder IP .

1.1. Intended audience

This document has been written for experienced hardware and software engineers who want to implement host applications by using the host interface registers.

1.2. Scope

This document mainly describes host interface registers that are used for communication between a host and the VPU(Video Processing Unit) such as host commands, VPU response, or temporal command arguments. It also covers interrupt and video operating control flow, and sample application codes using API functions.

1.3. Typographical conventions

The following typographical conventions are used in this document:

bold	Highlights signal names within text, and interface elements such as menu names. May also be used for emphasis in descriptive lists where appropriate.
italic	Highlights cross-references in blue, file names, and citations.
`typewriter`	Denotes example source codes and dumped character or text, data types, function, register, and flag names.

2. Further reading

This section lists documents which are related to this product.

2.1. Other documents

- *WAVE521 Datasheet*
- *WAVE521 Verification Guide*
- *WAVE521 API Reference Manual*

Glossary

ACLK	Clock source for AXI bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
AUD	Access Unit Delimiter
AVI	Audio Video Interleave, known by its acronym AVI, is a multimedia container format introduced by Microsoft
AVS2	The second generation AVS(Audio Video Coding Standard) primarily targeted for Ultra HD (High Definition) video applications.
AXI	Advanced eXtensible Interface. The ARM open standard for high-performance
BCLK	Clock source for V-CPU and BPU in V-CORE
BIT Processor	The name of C&M proprietary 16-bit DSP processor, dedicated to processing bitstream and controlling the video hardware blocks. Sometimes it is called just BIT in the document.
BPU	BIT Processor Unit
BW	Bandwidth
BWB	It stands for Burst Write Back. BWB is a hardware module that collects write data and send 8 bursts of data to the external memory. It aims to increase bus efficiency by reducing a lot of short burst accesses that happen from VPU due to the nature of CTU-based processing.
WTL	It represents Write To Linear. It is an optional hardware module that allows a reconstructed output to be written in linear frame as well as in compressed frame (if WTL is disabled, VPU only writes compressed frame for less bandwidth). WTL writes linear frame data by using the BWB module.
CABAC	Context-based Adaptive Binary Arithmetic Code
CBR	Constant Bit Rate
CCLK	Clock source for VCE in V-CORE
CFG	ConFiGuration with the cfg file extension
CIR	Committed Information Rate
CMD	Abbreviation for command. Commands are issued to VPU by Host processor through APB interface protocol.
C&M	Chips&Media
CODEC	COder DECoder for converting between analog and digital audio signals.

CTB	Coding Tree Block
CTU	Coding Tree Unit
CPB	Coded Picture Buffer, also known as Bitstream Buffer
DPB	Decoded Picture Buffer, also known as Frame Buffer
DSP	Digital Signal Processing
EVB	Evaluation
FBC	Frame Buffer Compressor hardware block
FBD	Frame Buffer Decompressor hardware block
FHD	Full High Definition (1080p; 1920x1080)
FIO	Fast IO, the internal bus interface for V-CPU
F/W	Firmware
GDI	General DMA Interface
GOB	Group of Blocks
GOP	Group of Picture
HD	High Definition (larger than SD size, in general 1280x720)
HEC	Header Extension Code
HEVC	High Efficiency Video Coding
HP	High Profile
HPI	Host Port Interface
IDR	Instantaneous Decoding Refresh
IP	Intellectual Property. It stands for the Chips&Media's video IP in this document.
ISO/IEC	ISO, the International Organization for Standardization, and IEC, the International Electrotechnical Commission
IRQ	Interrupt ReQuest
ISR	Interrupt Service Routine
ITU-T	International Telecommunication Union - Telecommunication Sector
KB	Kilo Byte
LSB	Least Significant Bit
MC	Motion Compensation
ME	Motion Estimation
MIPS	Millions of Instructions Per Second

MMF	MultiMedia Framework
MP	Main Profile
MPEG	Moving Picture Experts Group
MSB	Most Significant Bit
MV	Motion Vector
MVP	Motion Vector Predictor
NAL	Network Abstraction Layer
NB	Neighbor Block
PMV	Predicted Motion Vector
POC	Picture Order Count
PRP	The name of hardware unit for Pre-Processing
PPS	Picture Parameter Set
PU	Prediction Unit
QP	Quantization Parameter
RBSP	Raw Byte Sequence Payload
RDO	Rate Distortion Optimization
RS	Redundant Slice
RTP	Real-time Transfer Protocol
RESP	Abbreviation for response. Responses are written to indicate the operation result or status of VPU through APB interface protocol
SEI	Supplement Enhancement Information
SEQ	Sequence
SoC	System on Chip
SPS	Sequence Parameter Set
SSD	Sum of Squared Difference
SVC	Scalable Video Coding
SVAC	Surveillance Video and Audio Coding
UD	Ultra Definition (larger than FHD size)
4K UHD	4K Ultra High Definition (3840x2160)
8K UHD	8K Ultra High Definition (7680x3420)
VBR	Variable Bit Rate

VBV	Video Buffer Verifier
VCL	Video Coding Layer
VDI	VPU Device Driver Interface
VLC	Variable Length Code
VLD	Variable Length Decoder
VPU	Video Processing Unit. It stands for the Chips&Media's video IP in this document.
VPS	Video Parameter Set
VPUAPI	VPU Application Programming Interface consisting of a set of functions and data structures that programmers can use to create encode/decode application or interact with VPU
VP9	An open and royalty free video coding format developed by Google
V-CORE	Hardware codec implementation in VPU
V-CPU	32-bit Processor unit as top layer of VPU hierarchical architecture. It is responsible for parsing bitstream syntax from sequence to slice header unit, controlling the underlying video hardware blocks called V-CORE.

Chapter 1
HOST INTERFACE

1.1. VPU Control Scheme

This section presents general description of host interfaces that are provided for Host processor to control VPU(Video Processing Unit).

1.1.1. Communication Models

VPU requires two types of interfaces to communicate with Host processor.

Dedicated register interface

VPU requires a dedicated path for exchanging messages and information with Host processor. VPU uses a set of host interface registers for receiving a command from Host processor or sending a response to Host processor. The host interface registers can be accessed via AMBA APB (Advanced Peripheral Bus).

Shared memory

VPU also uses certain memory regions on SDRAM for storing data that is shared with Host processor. This kind of dedicated memory can hold bitstream data, frame data, and auxiliary data which are accessible through AMBA AXI bus by both VPU and Host processor.

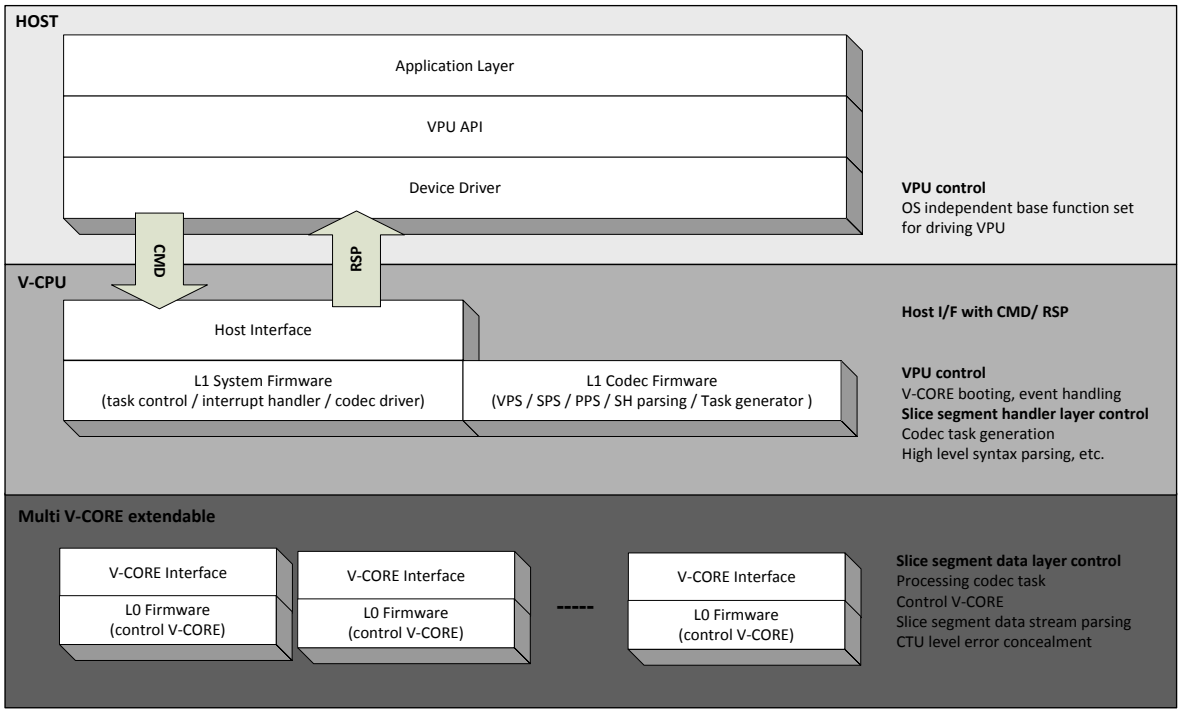


Figure 1.1. Exchanging Command/Response between Host and VPU

The host interface registers are classified into two groups: control registers and command registers.

- The control registers are mostly used to control VPU hardware and give the internal status or hardware information about VPU to Host processor.

- The command registers are used to issue commands and responses for video codec processing. Basically, VPU provides a set of pre-defined commands and their corresponding responses. L1 firmware, which runs on VPU, is designed for these given sets of commands and responses.

Host processor and VPU can access directly all bitstream data, frame data, and auxiliary data in SDRAM via AXI bus. The related information about all those data transfers are exchanged on host interface register via commands and responses. In order to do this, VPU provides a set of registers accessible from Host processor.

Generally, all of these transactions are one-directional transactions, which means one only writes data and the other only reads the written data on a single data buffer like stream buffer case to assure safe transactions between Host processor and VPU.

VPU works with the commands that are queued by Host processor. After completion of the job, VPU returns the result of the command or requires other information on the host interface register. VPU only can manage frame buffers associated with picture decoding.

Besides the frame buffer and stream buffer, VPU requires secured memory regions for video processing, which are called a Code Buffer, Work Buffer and Temporal Buffer. They are used for manipulation of firmware code, static data and temporal data respectively. These buffers are only accessible by VPU.

Note There are restrictions on allocation of the buffer address. The base buffer addresses must be aligned in a 4KB unit. Also, 0xFFFE0000 to 0xFFFFFFFF (based on 32-bit) area is mapped to FIO (VCPU internal bus), so Host processor must not use this area for the buffer addresses.

1.2. Host Interface Registers

1.2.1. Overview of Host Interface Registers

Host processor and VPU can exchange data on host interface registers that are memory-mapped through APB. The following table shows the range of APB offsets for access to VPU.

Table 1.1. APB Offsets for VPU

APB start	APB end	Region
0x0000	0x00FF	Host interface register - control registers
0x0100	0x01FF	Host interface register - command I/O registers
0x0200	0x0FFF	Reserved

Control Registers

Host interface registers in this category (0x0000 to 0x00FF) are used to control VPU hardware or to show the VPU status. Host processor can use most of these registers for initializing VPU during booting process.

Command I/O Registers

Host interface registers in this category (0x0100 to 0x01FF) are overwritten or updated whenever a new command is given to VPU from Host processor. All the commands with input arguments and all the corresponding responses from VPU are delivered through these registers.

In fact, the command register region is shared for parameters of the commands. There are many predefined commands as described in *Section 1.2.2.3, "Host Commands"*. Most of the commands have the same address of parameter registers in common. However, the same address of register might have a different meaning depending on the command type that has been issued by Host processor. For example, the base address + 0x0100 is always a run command register. However, the base address + 0x0134 means the number of task buffer for INIT_VPU command, while it means the base address of luma frame buffer for SET_FB command.

For information on the list of the command I/O registers, please see the *Section 1.2.3.2, "Summary of Command I/O registers"*.

Figure 1.2, "Host Interface Memory Map" represents the APB connected memory map for host's access to VPU.

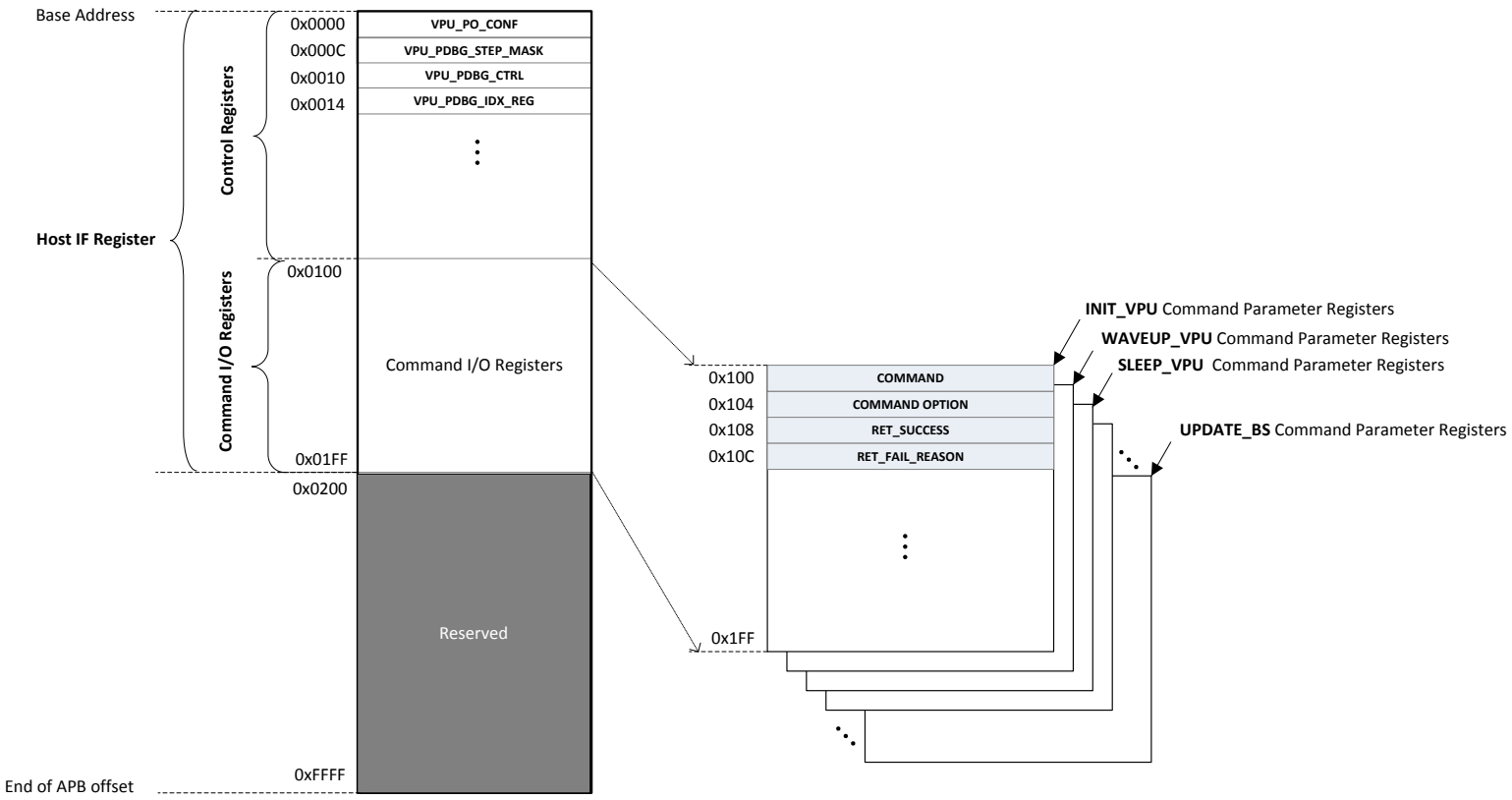


Figure 1.2. Host Interface Memory Map

1.2.2. Command Interface Overview

Host processor can give any command to VPU by setting COMMAND register(0x100) and also can send an interrupt request to VPU for the command issued through VPU_HOST_INT_REQ register(0x038). Before writing to host interface register, Host processor should check the ownership of host interface registers.

1.2.2.1. Ownership of Host Interface Registers

VPU_BUSY_STATUS(0x070) register indicates which side between VPU and Host processor has the ownership of access to host interface registers.

- 1 of VPU_BUSY_STATUS : VPU has the ownership for access to host interface registers.
- 0 of VPU_BUSY_STATUS : Host processor has the ownership for access to host interface registers.

Host processor must check the ownership through the VPU_BUSY_STATUS register prior to sending command and arguments. Host processor can send a command when VPU_BUSY_STATUS is 0. After setting the command arguments, Host processor must set the VPU_BUSY_STATUS register to 1 and issue the command to give the access ownership to VPU. When VPU_BUSY_STATUS turns 0 again, Host processor can issue another command for secure operation.

1.2.2.2. Command Protocol

WAVE5 series VPU supports command-queueing to maximize performance by pipelining internal commands and by hiding wait cycle taken to receive a command from Host processor.

Command Queueing

In the frame-by-frame basis command protocol, VPU should wait for the next command. Also it is hard to get advantage from command level pipelining.

WAVE5 series VPU adopts command queueing scheme to improve command protocol performance. In command queueing scheme, VPU has two queues. One is a command-queue for queueing commands from Host processor, and the other is a report-queue for queueing the results of the commands.

Host-issued commands are queued into the command-queue in order. VPU gets the commands which are kept in the command-queue and works for the commands one by one. VPU writes a return value or command result into the report-queue. Host processor and VPU can run in parallel with minimum intervention by using the command queue architecture.

Host Processor

Instead of waiting for each command to be executed before sending the next command, Host processor just places all the commands in the command-queue and goes on doing other things while the commands in the queue are processed by VPU.

While Host processor handles its own tasks, it can receive VPU IRQ(Interrupt ReQuest). In this case, Host processor can simply exit ISR(Interrupt Service Routine) without access to host interface registers to read the result of the command reported by VPU. After Host processor completes its tasks, Host processor can read the command result when Host processor needs the reports and does response processing.

VPU

To start next frame decoding/encoding, VPU fetches a command from the command-queue, and runs decoder/encoder pipeline without any host triggering. When the command is done, VPU stores the command result into the report-queue and goes on for the next frame.

Figure 1.3, "VPU's Internal Elastic Pipelines in Command Queue Architecture" presents Host processor and internal pipelines of VPU in WAVE5 command architecture.

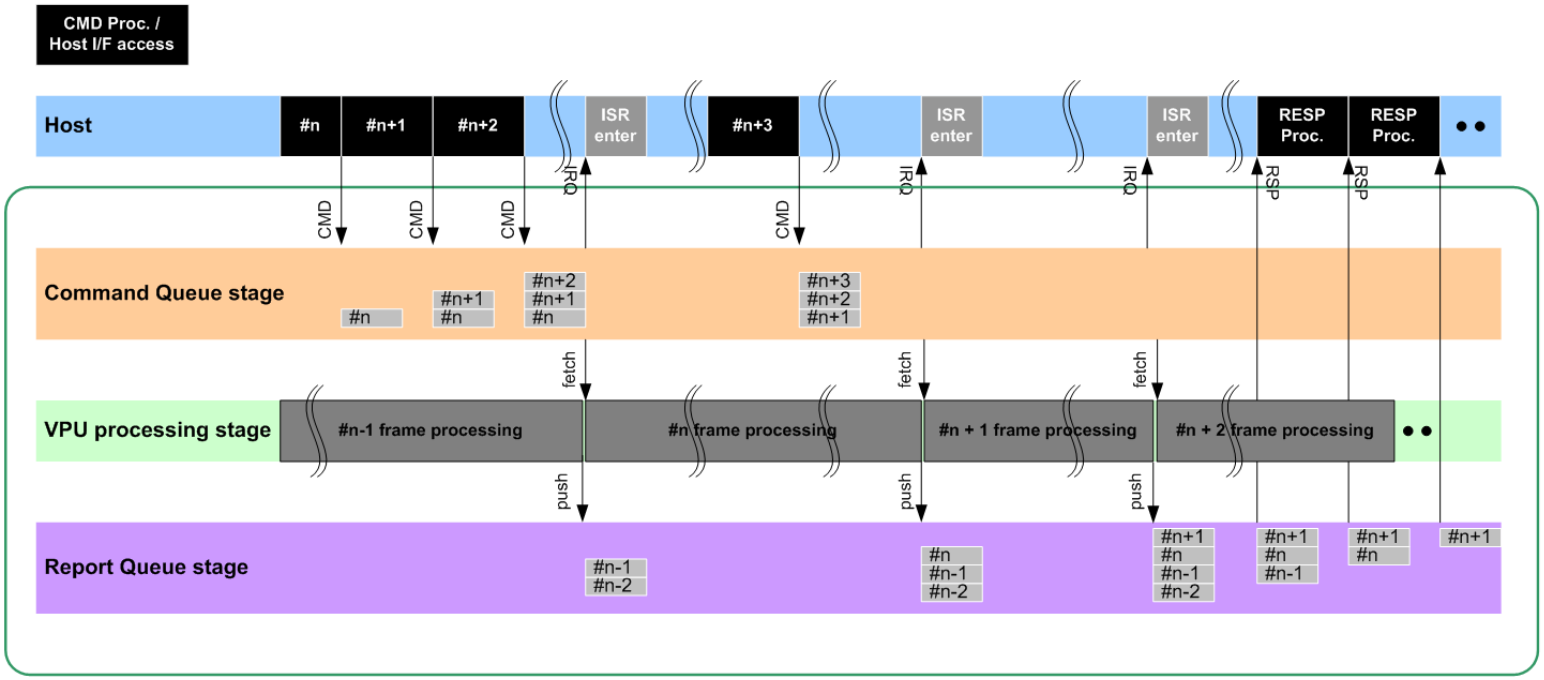


Figure 1.3. VPU's Internal Elastic Pipelines in Command Queue Architecture

Host processor can continue to send commands until the command-queue gets full. It also does not have to do immediately response processing when ISR is called. In other words, it can handle all the responses at once by using QUERY command (to be explained in the following chapter).

Meantime, VPU is able to keep more than one command in the command-queue. In the command queueing, VPU can execute commands in a parallel pipeline fashion. Moreover, VPU can run tasks without any intervention by Host processor when there are enough commands in the command-queue and enough room in the report-queue.

Queueing Commands to VPU

As previously described, VPU can send command parameters and receive response results through host interface register. The VPU_BUSY_STATUS register (0x070) indicates the ownership of host interface register. Host processor should do the following things with regard to the VPU_BUSY_STATUS register between sending commands.

1. Host processor should check whether VPU_BUSY_STATUS is 0 before sending a command to the command-queue.

2. After setting the command arguments, Host processor should change VPU_BUSY_STATUS to 1 and issue the command.

3. If VPU has done with the CQ required task for the command (even with occurrance of error), it changes VPU_BUSY_STATUS to 0. It indicates completion of the job for command queue.

Host processor should keep in mind that 0 of VPU_BUSY_STATUS does not mean that VPU is in idle state. It just indicates that VPU has no access ownership to host interface register while VPU_BUSY_STATUS is 0.

Host processor can send a command even when the command-queue is full. However, in that case VPU returns fail for the not-queued command and Host processor should try the command again.

Querying Results from VPU

There is a QUERY command to get the result of command that has been executed by VPU. Host processor can send QUERY command and get reported whenever Host processor wants. In most cases, after the execution of the given command, VPU writes the command result to report-queue. Then VPU raises an interrupt and proceeds to work with a next command in the command-queue.

If there are more than one command in the report-queue, Host processor can retrieve them one by one by calling QUERY commands continuously.

Host processor can give the QUERY command even when report-queue is empty. However, VPU returns fail and Host processor should try the command again.

1.2.2.3. Host Commands

VPU has three kinds of predefined command sets. In this section, we'll describe the command sets briefly.

Table 1.2. Command Sets

Command Set Type	Command	Description
Commands for VPU hardware control	INIT_VPU	This command initializes VPU. Host processor must call the command after hardware reset.
	SLEEP_VPU	This command makes VPU go into sleep status.
	WAVE_VPU	This command wakes up VPU from sleep status.
Sequence level commands	CREATE_INST	This command allocates memory for the instance.
	FLUSH_INST	This command finalizes instance with flushing all the information.
	DISTORY_INST	This command deallocates memory for the instance.
	INIT_SEQ	This command initializes a video sequence for decoding. It returns sequence level information. In general, this information can be used for allocating frame buffer.
	SET_PARAM	This command sets initial encoding parameters such as the size of source and GOP structures.
	SET_FB	This command sets and allocates the required frame buffer memory space.
Frame level commands	DEC_PIC/ ENC_PIC	This command decodes or encodes a frame.
	SET_PARAM (CHANGE_PARAM)	This command changes encoding paramters which affect only a frame, not the whole sequence. CHANGE_PARAM is a part of SET_PARAM command.
	QUERY	This command queries the result of command execution.
	UPDATE_BS	This command updates bitstream buffer. In general, read pointer (encoder case) or write pointer (decoder case) is updated by this command.

1.2.2.3.1. Queueable and Non-queueable Commands

The WAVE5 host command is mainly composed of non-queueable commands and queueable commands.

Table 1.3. Command Classification by Command Queue

Command Set Type	Command
Queueable host command	INIT_SEQ
	DEC_PIC
	ENC_PIC

Command Set Type	Command
	SET_PARAM
Non-queueable host command	INIT_VPU
	SLEEP_VPU
	WAKE_VPU
	CREATE_INST
	FLUSH_INST
	SET_FB
	QUERY
	UPDATE_BS

Queueable commands are command sets which have something to do with parsing and encoding/decoding operation. Initially these commands are sent to the command-queue. After executing the commands one by one, VPU saves the results in the report-queue.

Non-queueable commands are mostly the ones that are related to initialization and termination of VPU itself. For the reason, they are not managed in the command-queue.

1.2.2.3.2. SLEEP_VPU and WAKE_VPU

VPU has some restriction on the SLEEP_VPU and WAKE_VPU command. VPU can perform sleep and wake operation only when queueing commands are all flushed and VPU is in idle state.

To satisfy such conditions, Host processor first needs to check the state of command queue before sending the SLEEP_VPU and WAKE_VPU command to VPU.

1.2.2.3.3. Trick Play during Decoding

Trick Play is a video function that allows a subset of frames to be presented in decoder such as fast-forward play. Host processor can enable Trick Play by sending DEC_PIC command with enabling Thumbnail mode. Also, there are picture skip related functions such as skip non-IRAP, skip non-I picture, or skip non-ref frame. The picture skip can be enabled by setting the register CMD_DEC_PIC_OPTION (0x104).

When Trick Play option is turned on, VPU internally sets the command-queue depth to 1. For enabling Trick Play decoding option, the command-queue should be empty. VPU returns fail if another command already exists in the command-queue. Then Host processor should retry to do the trick-play.

There are two ways to get the command-queue to be empty.

1. Before trick-play, Host processor gives a flash option which makes the queued commands nullify.

2. Host processor waits for execution of all commands to be finished.

In order to disable display reordering and to display decoded frames immediately while trick-play is on, Host processor should set CMD_DEC_FORCE_FB_LATENCY_PLUS1 (0x134) to 1. VPU always turns off display reordering when thumbnail mode is on. If Host processor wants to resume normal decoding from the current frame, it should disable trick-play option and set CMD_DEC_FORCE_FB_LATENCY_PLUS1 to 0.

1.2.2.3.4. Setting and Changing Encoding Parameters

Host processor can give encoding parameters to VPU by using ENC_SET_PARAM command. There are three types of ENC_SET_PARAM command depending on SET_PARAM_OPTION: COMMON, GOP, and CHANGE_PARAM.

The COMMON and GOP are sequence-level parameters, so they are set only once for the instance. The CHANGE_PARAM is the paramters that can be changed for each frame. Host processor can change picture-level parameters or rate control parameters. However, SPS relevant parameters are not allowed to be changed with CHANGE_PARAM.

For the detailed description, please refer to *Section 1.2.3.2.8, "ENC_SET_PARAM(COMMON) Command Parameter Registers"*.

1.2.2.3.5. Interrupt Interface

Host-asserted Interrupt

After sending a video command or handling bitstream, Host processor can raise an interrupt that informs VPU of the status change. To give an interrupt to VPU,

1. Host processor sets the command on COMMAND register(0x100) and the relevant parameters for the command to Command I/O registers.

2. Host processor writes 1 to VPU_HOST_INT_REQ register(0x038).

Before giving an interrupt, VPU_BUSY_STATUS register (0x070) must be set to 1. While VPU handles the interrupt, VPU_HOST_INT_REQ(0x038) turns 0.

VPU-asserted Interrupt

VPU can send Host processor an interrupt to show the result of command execution and to require more bitstream in the bistream buffer. The following describes the situation when VPU raises an interrupt.

1. If VPU meets the case that the hardware interrupt needs to be triggered, it checks the value of bitfield on VPU_VINT_ENABLE register (0x048) which indicates each of interrupt source. If any of those bitfields is set to 1(ENABLE), the corresponding interrupt can occur.

2. VPU sets o_vpu_intrpt signal to 1, which issues the interrupt. Then VPU_VPU_INT_STS register (0x044) turns 1.

3. Depending on the source of interrupt, the relevant bitfield of VPU_VINT_REASON register (0x04C) becomes 1. At the same time, the relevant bitfield of VPU_VINT_REASON_USER register (0x030) also becomes 1.

Host processor checks the value of VPU_VINT_REASON in ISR, schedules the relevant service, and clears the interrupt as the following procedure.

1. Host processor clears the interrupt reason by writing 1 to the interrupt source bitfield of VPU_VINT_REASON_CLR register (0x034).

2. Host processor clears the interrupt by setting 1 to VPU_VINT_CLEAR register (0x03C).

3. VPU_VPU_INT_STS register (0x044) value is automatically changed to 0. The o_vpu_intrpt value also turns 0.

4. If VPU_VINT_REASON register is not still 0 (because another interrupt arises and pending while the previous interrupt is being cleared), the new interrupt is issued after 1 cycle.

VPU_VINT_REASON_USER register is not affected by VPU_VINT_REASON_CLR register. The VPU_VINT_REASON_USER register keeps the interrupt reasons that has previously occurred. It helps Host processor confirm which kind of interrupt has happened. Host processor should need explicit action to clear the interrupt by writing the VPU_VINT_REASON_USER to 0.

1.2.3. Summary of Host Interface Registers

1.2.3.1. Summary of Control Registers

Table 1.4. Summary of Control Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000000	RW	0x0	*VPU_PO_CONF*	Power On Configurations
0x0000000C	RW	0x0	*VPU_PDBG_STEP_MASK*	V-CPU Debugger Step Mask
0x00000010	RW	0x0	*VPU_PDBG_CTRL*	V-CPU Debugger Control
0x00000014	RW	0x0	*VPU_PDBG_IDX_REG*	V-CPU Debugger Index
0x00000018	RW	0x0	*VPU_PDBG_WDATA_REG*	V-CPU Debugger Write Data
0x00000020	RW	0x0	*VPU_FIO_CTRL_ADDR*	FastIO Control/Address
0x00000024	RW	0x0	*VPU_FIO_DATA*	FastIO Data
0x00000034	WO	0x0	*VPU_VINT_REASON_CLR*	Interrupt Reason Clear
0x00000038	WO	0x0	*VPU_HOST_INT_REQ*	Host Interrupt Request
0x0000003C	WO	0x0	*VPU_VINT_CLEAR*	VPU Interrupt Clear
0x00000048	RW	0x0	*VPU_VINT_ENABLE*	VPU Interrupt Enable
0x0000004C	RW	0x0	*VPU_VINT_REASON*	VPU Interrupt Reason
0x00000050	RW	0x0	*VPU_RESET_REQ*	VPU Reset Request
0x00000058	RW	0x0	*VCPU_RESTART*	V-CPU Restart Request
0x0000005C	RW	0x0	*VPU_CLK_MASK*	VPU Clock Control
0x00000060	RW	0x0	*VPU_REMAP_CTRL*	Remap Control
0x00000064	RW	0x0	*VPU_REMAP_VADDR*	Remap Virutal Address
0x00000068	RW	0x0	*VPU_REMAP_PADDR*	Remap Physical Address
0x0000006C	RW	0x0	*VPU_REMAP_CORE_START*	VPU Start Request
0x00000070	RW	0x0	*VPU_BUSY_STATUS*	VPU Busy Status
0x00000300	RW	0x0	*ENC_PIC_SUB_FRAME_SYNC_IF*	Subframe Sync register interface
OUTPUT RETURN				
0x00000004	RW	0x0	*VCPU_CUR_PC*	Current PC
0x00000008	RW	0x0	*VCPU_CUR_LR*	Current RL
0x0000001C	RW	0x0	*VPU_PDBG_RDATA_REG*	V-CPU Debugger Read Data
0x00000030	RW	0x0	*VPU_VINT_REASON_USR*	Interrupt Reason User
0x00000040	RO	0x0	*VPU_HINT_CLEAR*	Host Interrupt Clear
0x00000044	RO	0x0	*VPU_VPU_INT_STS*	VPU Interrupt Status
0x00000054	RW	0x0	*VPU_RESET_STATUS*	VPU Reset Status
0x00000074	RW	0x0	*VPU_HALT_STATUS*	VPU Halt Status
0x00000078	RW	0x0	*VPU_VCPU_STATUS*	N/A
0x0000007C	RW	0x0	*RSVD*	RSVD
0x00000080	RW	0x0	*RET_FIO_STATUS*	RET_FIO_STATUS
0x00000090	RW	0x0	*RET_PRODUCT_NAME*	HW product name
0x00000094	RW	0x0	*RET_PRODUCT_VERSION*	HW product version
0x00000098	RW	0x0	*RET_VCPU_CONFIG0*	Configuration Information #0
0x0000009C	RW	0x0	*RET_VCPU_CONFIG1*	Configuration Information #1
0x000000A0	RW	0x0	*RET_CODEC_STD*	Standard Definition
0x000000A4	RW	0x0	*RET_CONF_DATE*	Configuration Date

Offset	Type	Reset Value	Name	Description
0x000000A8	RW	0x0	*RET_CONF_REVISION*	The revision of H/W configuration
0x000000AC	RW	0x0	*RET_CONF_TYPE*	The define value of H/W configuration
0x000000B0	RW	0x0	*RET_VCORE0_CFG*	Configuration Information of VCORE0
0x000000B4	RW	0x0	*RET_VCORE1_CFG*	Configuration Information of VCORE1
0x000000B8	RW	0x0	*RET_VCORE2_CFG*	Configuration Information of VCORE2
0x000000BC	RW	0x0	*RET_VCORE3_CFG*	Configuration Information of VCORE3
0x000000D0	RW	0x0	*VPU_RET_VCORE_PRESET*	Number of VCOREs present

1.2.3.2. Summary of Command I/O registers

Among host interface registers, Command I/O Registers are used in a pre-defined way for each command controlling VPU. Sample usage of these Command I/O registers can be summarized as the following sub-sections.

1.2.3.2.1. Common Parameter Registers

Table 1.5. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Command
0x00000104	RW	0x0	*CMD_OPTION*	Command Option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Result of the command
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail reason of the run command
0x000001E0	RW	0x0	*RET_QUEUE_STATUS*	Queued command information
0x000001E4	RW	0x0	*RET_BS_EMPTY*	Bitstream buffer empty flag
0x000001E8	RW	0x0	*RET_QUEUED_CMD_DONE*	Queued command done flag
0x000001EC	RW	0x0	*RET_SEEK_INSTANCE_INFO*	A working instance index on seek stage (for internal use only)
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	A working instance index on prescan stage (for internal use only)
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	A working instance index on decoding stage (for internal use only)
0x000001F8	RW	0x0	*RET_ENCODING_INSTANCE_INFO*	A working instance index on packing stage (for internal use only)
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Picture command done interrupt instance index

1.2.3.2.2. INIT_VPU Command Parameter Registers

Table 1.6. INIT_VPU Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000110	RW	0x0	*ADDR_CODE_BASE*	Code buffer base address
0x00000114	RW	0x0	*CODE_SIZE*	Code buffer size
0x00000118	RW	0x0	*CODE_PARAM*	Code buffer parameters
0x0000011C	RW	0x0	*CMD_INIT_ADDR_TEMP_BASE*	Temporal buffer base address
0x00000120	RW	0x0	*CMD_INIT_TEMP_SIZE*	Temporal buffer size

Offset	Type	Reset Value	Name	Description
0x00000124	RW	0x0	*CMD_INIT_ADDR_SEC_AXI*	Secondary AXI base address
0x00000128	RW	0x0	*CMD_INIT_SEC_AXI_SIZE*	Seconary AXI memory size
0x0000012C	RW	0x0	*CMD_INIT_HW_OPTION*	VPU hardware option
0x00000130	RW	0x0	*CMD_WAKEUP_SYSTEM_CLOCK*	Time out count
0x00000134	RW	0x0	*CMD_INIT_NUM_TASK_BUF*	Number of task buffer
0x00000138	RW	0x0	*CMD_INIT_ADDR_TASK_BUF0*	Base address of task buffer 0
0x0000013C	RW	0x0	*CMD_INIT_ADDR_TASK_BUF1*	Base address of task buffer 1
0x00000140	RW	0x0	*CMD_INIT_ADDR_TASK_BUF2*	Base address of task buffer 2
0x00000144	RW	0x0	*CMD_INIT_ADDR_TASK_BUF3*	Base address of task buffer 3
0x00000148	RW	0x0	*CMD_INIT_ADDR_TASK_BUF4*	Base address of task buffer 4
0x0000014C	RW	0x0	*CMD_INIT_ADDR_TASK_BUF5*	Base address of task buffer 5
0x00000150	RW	0x0	*CMD_INIT_ADDR_TASK_BUF6*	Base address of task buffer 6
0x00000154	RW	0x0	*CMD_INIT_ADDR_TASK_BUF7*	Base address of task buffer 7
0x00000158	RW	0x0	*CMD_INIT_ADDR_TASK_BUF8*	Base address of task buffer 8
0x0000015C	RW	0x0	*CMD_INIT_ADDR_TASK_BUF9*	Base address of task buffer 9
0x00000160	RW	0x0	*CMD_INIT_ADDR_TASK_BUFA*	Base address of task buffer A
0x00000164	RW	0x0	*CMD_INIT_ADDR_TASK_BUFB*	Base address of task buffer B
0x00000168	RW	0x0	*CMD_INIT_ADDR_TASK_BUFC*	Base address of task buffer C
0x0000016C	RW	0x0	*CMD_INIT_ADDR_TASK_BUFD*	Base address of task buffer D
0x00000170	RW	0x0	*CMD_INIT_ADDR_TASK_BUFE*	Base address of task buffer E
0x00000174	RW	0x0	*CMD_INIT_ADDR_TASK_BUFF*	Base address of task buffer F
0x00000178	RW	0x0	*CMD_INIT_TASK_BUFF_SIZE*	Size of task buffer
OUTPUT RETURN				

1.2.3.2.3. WAKEUP_VPU Command Parameter Registers

Table 1.7. WAKEUP_VPU Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000110	RW	0x0	*CMD_WAKEUP_ADDR_CODE_BASE*	Code buffer base address
0x00000114	RW	0x0	*CMD_WAKEUP_CODE_SIZE*	Code buffer size
0x00000118	RW	0x0	*CMD_WAKEUP_CODE_PARAM*	Code buffer parameter
0x0000011C	RW	0x0	*CMD_WAKEUP_ADDR_TEMP_BASE*	Temporal buffer base address
0x00000120	RW	0x0	*CMD_WAKEUP_TEMP_SIZE*	Temporal buffer size
0x00000124	RW	0x0	*CMD_WAKEUP_ADDR_SEC_AXI*	Secondary AXI base address
0x00000128	RW	0x0	*CMD_WAKEUP_SEC_AXI_SIZE*	Seconary AXI memory size
0x0000012C	RW	0x0	*CMD_WAKEUP_HW_OPTION*	VPU hardware option
0x00000130	RW	0x0	*CMD_WAKEUP_SYSTEM_CLOCK*	Time out count
0x00000134	RW	0x0	*CMD_WAKEUP_NUM_TASK_BUF*	Number of task buffer
0x00000138	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF0*	Base address of task buffer 0
0x0000013C	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF1*	Base address of task buffer 1
0x00000140	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF2*	Base address of task buffer 2
0x00000144	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF3*	Base address of task buffer 3
0x00000148	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF4*	Base address of task buffer 4
0x0000014C	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF5*	Base address of task buffer 5
0x00000150	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF6*	Base address of task buffer 6

Offset	Type	Reset Value	Name	Description
0x00000154	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF7*	Base address of task buffer 7
0x00000158	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF8*	Base address of task buffer 8
0x0000015C	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUF9*	Base address of task buffer 9
0x00000160	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUFA*	Base address of task buffer A
0x00000164	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUFB*	Base address of task buffer B
0x00000168	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUFC*	Base address of task buffer C
0x0000016C	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUFD*	Base address of task buffer D
0x00000170	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUFE*	Base address of task buffer E
0x00000174	RW	0x0	*CMD_WAKEUP_ADDR_TASK_BUFF*	Base address of task buffer F
0x00000178	RW	0x0	*CMD_WAKEUP_TASK_BUFF_SIZE*	Size of task buffer
OUTPUT RETURN				

1.2.3.2.4. SLEEP_VPU Command Parameter Registers

Table 1.8. SLEEP_VPU Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run command
OUTPUT RETURN				
0x00000110	RW	0x0	*RET_SUCCESS*	Run command status
0x00000114	RW	0x0	*RET_FAIL_REASON*	Fail reasons
0x00000124	RW	0x0	*RET_CUR_SP*	Return Current Stack Pointer

1.2.3.2.5. CREATE_INST Command Parameter Registers for Encoder

Table 1.9. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000114	RW	0x0	*CMD_CREATE_INST_ADDR_WORK_BASE*	Work buffer base address
0x00000118	RW	0x0	*CMD_CREATE_INST_WORK_SIZE*	Work buffer size
0x00000128	RW	0x0	*CMD_CREATE_INST_SUB_FRM_SYNC*	Enable subframe synchronization
OUTPUT RETURN				

1.2.3.2.6. FLUSH_INST Command Parameter Registers

Table 1.10. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run command
0x00000104	RW	0x0	*CMD_CREATE_INST_OPTION*	Run command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Result of the command
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail reason of the run command
0x000001EC	RW	0x0	*RET_SEEK_INSTANCE_INFO*	A working instance index on seek stage (for internal use only)

Offset	Type	Reset Value	Name	Description
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	A working instance index on prescan stage (for internal use only)
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	A working instance index on decoding stage (for internal use only)
0x000001F8	RW	0x0	*RET_ENCODING_INSTANCE_INFO*	A working instance index on packing stage (for internal use only)

1.2.3.2.7. DESTROY_INST Command Parameter Registers

Table 1.11. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run command
0x00000104	RW	0x0	*CMD_CREATE_INST_OPTION*	Run command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Result of the command
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail reason of the run command
0x000001EC	RW	0x0	*RET_SEEK_INSTANCE_INFO*	A working instance index on seek stage (for internal use only)
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	A working instance index on prescan stage (for internal use only)
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	A working instance index on decoding stage (for internal use only)
0x000001F8	RW	0x0	*RET_ENCODING_INSTANCE_INFO*	A working instance index on packing stage (for internal use only)

1.2.3.2.8. ENC_SET_PARAM(COMMON) Command Parameter Registers

Table 1.12. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000104	RW	0x0	*SET_PARAM_OPTION*	ENC_SET_PARAM command option
0x00000118	RW	0x0	*CMD_ENC_SET_PARAM_ENABLE*	Encoder parameter change setting
0x0000011C	RW	0x0	*CMD_ENC_SEQ_SRC_SIZE*	A size of source picture
0x00000120	RW	0x0	*CMD_ENC_SEQ_CUSTOM_MAP_ENDIAN*	Custom map endianness
0x00000124	RW	0x0	*CMD_ENC_SEQ_SPS_PARAM*	HEVC encoder sequence parameters
0x00000128	RW	0x0	*CMD_ENC_SEQ_PPS_PARAM*	HEVC encoder picture parameters
0x0000012C	RW	0x0	*CMD_ENC_SEQ_GOP_PARAM*	GOP parameters
0x00000130	RW	0x0	*CMD_ENC_SEQ_INTRA_PARAM*	Intra picture coding parameters
0x00000134	RW	0x0	*CMD_ENC_SEQ_CONF_WIN_TOP_BOT*	Top and bottom size for conformance window
0x00000138	RW	0x0	*CMD_ENC_SEQ_CONF_WIN_LEFT_RIGHT*	Left and right size for conformance window
0x0000013C	RW	0x0	*CMD_ENC_SEQ_RDO_PARAM*	RDO coding options
0x00000140	RW	0x0	*CMD_ENC_SEQ_INDEPENDENT_SLICE*	Slice parameters for an independent slice segment
0x00000144	RW	0x0	*CMD_ENC_SEQ_DEPENDENT_SLICE*	Slice parameters for a dependent slice segment
0x00000148	RW	0x0	*CMD_ENC_SEQ_INTRA_REFRESH*	Intra refresh mode

Offset	Type	Reset Value	Name	Description
0x0000014C	RW	0x0	CMD_ENC_SEQ_INPUT_SRC_PARAM	Source frame parameter
0x00000150	RW	0x0	CMD_ENC_RC_FRAME_RATE	Frame rate for rate control
0x00000154	RW	0x0	CMD_ENC_RC_TARGET_RATE	Target bitrate
0x00000158	RW	0x0	CMD_ENC_RC_PARAM	Rate control parameters
0x0000015C	RW	0x0	CMD_ENC_RC_MIN_MAX_QP	Min/Max QP for rate control
0x00000160	RW	0x0	CMD_ENC_RC_BIT_RATIO_LAYER_0_3	RC_BIT_RATIO_LAYER_0_3
0x00000164	RW	0x0	CMD_ENC_RC_BIT_RATIO_LAYER_4_7	RC_BIT_RATIO_LAYER_4_7
0x00000168	RW	0x0	CMD_ENC_RC_INTER_MIN_MAX_QP	Min/Max QP for rate control
0x0000016C	RW	0x0	CMD_ENC_SEQ_RESERVED	Reserved
0x00000170	RW	0x0	CMD_ENC_ROT_PARAM	Rotation and mirror mode
0x00000174	RW	0x0	CMD_ENC_NUM_UNITS_IN_TICK	NUM_UNITS_IN_TICK
0x00000178	RW	0x0	CMD_ENC_TIME_SCALE	TIME_SCALE
0x0000017C	RW	0x0	CMD_ENC_NUM_TICKS_POC_DIFF_ONE	NUM_TICKS_POC_DIFF_ONE
0x00000184	RW	0x0	CMD_ENC_CUSTOM_MD_PU04	Custome mode decision for PU04
0x00000188	RW	0x0	CMD_ENC_CUSTOM_MD_PU08	Custome mode decision for PU08
0x0000018C	RW	0x0	CMD_ENC_CUSTOM_MD_PU16	Custome mode decision for PU16
0x00000190	RW	0x0	CMD_ENC_CUSTOM_MD_PU32	Custome mode decision for PU32
0x00000194	RW	0x0	CMD_ENC_CUSTOM_MD_CU08	Custome mode decision for CU08
0x00000198	RW	0x0	CMD_ENC_CUSTOM_MD_CU16	Custome mode decision for CU16
0x0000019C	RW	0x0	CMD_ENC_CUSTOM_MD_CU32	Custome mode decision for CU32
0x000001A0	RW	0x0	CMD_ENC_NR_PARAM	Noise reduction parameter
0x000001A4	RW	0x0	CMD_ENC_NR_WEIGHT	Noise reduction weight parameter
0x000001A8	RW	0x0	CMD_ENC_BG_PARAM	Background encoding parameter
0x000001AC	RW	0x0	CMD_ENC_CUSTOM_LAMBDA_ADDR	Custom lambda map address
0x000001B0	RW	0x0	CMD_ENC_USER_SCALING_LIST_ADDR	User scaling list address
OUTPUT RETURN				

1.2.3.2.9. ENC_SET_PARAM(GOP) Command Parameter Registers

Table 1.13. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000104	RW	0x0	SET_PARAM_OPTION	ENC_SET_PARAM command option
0x0000011C	RW	0x0	CMD_ENC_CUSTOM_GOP_PARAM	Size of source pictures of custom GOP
0x00000120	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_0	Parameters for the 0th picture of custom GOP
0x00000124	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_1	Parameters for the 1st picture of custom GOP
0x00000128	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_2	Parameters for the 2nd picture of custom GOP
0x0000012C	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_3	Parameters for the 3rd picture of custom GOP
0x00000130	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_4	Parameters for the 4th picture of custom GOP
0x00000134	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_5	Parameters for the 5th picture of custom GOP
0x00000138	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_6	Parameters for the 6th picture of custom GOP

Offset	Type	Reset Value	Name	Description
0x0000013C	RW	0x0	CMD_ENC_CUSTOM_GOP_PIC_PARAM_7	Parameters for the 7th picture of custom GOP
OUTPUT RETURN				

1.2.3.2.10. SET_FB Command Parameter Registers for Encoder

Table 1.14. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000104	RW	0x0	CMD_SET_FB_OPTION	SET_FB command option
0x00000118	RW	0x0	CMD_SET_FB_COMMON_PIC_INFO	DPB information
0x0000011C	RW	0x0	CMD_SET_FB_PIC_SIZE	Decoded picture size
0x00000120	RW	0x0	CMD_SET_FB_NUM_FB	The number of start/end frame buffer to set
0x00000128	RW	0x0	CMD_SET_FB_FBC_STRIDE	Frame buffer setting for compressed frame
0x0000012C	RW	0x0	CMD_SET_FB_ADDR_SUB_SAMPLED_FB_BASE	Base address of frame buffer for sub-sampled frame
0x00000130	RW	0x0	CMD_SET_FB_SUB_SAMPLED_ONE_FB_SIZE	Size of frame buffer for sub-sampled frame
0x00000134	RW	0x0	CMD_SET_FB_ADDR_LUMA_BASE0	Luma base of index0
0x00000138	RW	0x0	CMD_SET_FB_ADDR_CB_BASE0	Cb base of index0
0x0000013C	RW	0x0	CMD_SET_FB_ADDR_CR_BASE0	Cr base of index0
0x0000013C	RW	0x0	CMD_SET_FB_ADDR_FBC_Y_OFFSET0	FBC luma offset base of index0
0x00000140	RW	0x0	CMD_SET_FB_ADDR_FBC_C_OFFSET0	FBC chroma offset base of index0
0x00000144	RW	0x0	CMD_SET_FB_ADDR_LUMA_BASE1	Luma base of index1
0x00000148	RW	0x0	CMD_SET_FB_ADDR_CB_BASE1	Cb base of index1
0x0000014C	RW	0x0	CMD_SET_FB_ADDR_CR_BASE1	Cr base of index1
0x0000014C	RW	0x0	CMD_SET_FB_ADDR_FBC_Y_OFFSET1	FBC luma offset base of index1
0x00000150	RW	0x0	CMD_SET_FB_ADDR_FBC_C_OFFSET1	FBC chroma offset base of index1
0x00000154	RW	0x0	CMD_SET_FB_ADDR_LUMA_BASE2	Luma base of index2
0x00000158	RW	0x0	CMD_SET_FB_ADDR_CB_BASE2	Cb base of index2
0x0000015C	RW	0x0	CMD_SET_FB_ADDR_CR_BASE2	Cr base of index2
0x00000160	RW	0x0	CMD_SET_FB_ADDR_FBC_C_OFFSET2	FBC chroma offset base of index2
0x00000164	RW	0x0	CMD_SET_FB_ADDR_LUMA_BASE3	Luma base of index3
0x00000168	RW	0x0	CMD_SET_FB_ADDR_CB_BASE3	Cb base of index3
0x0000016C	RW	0x0	CMD_SET_FB_ADDR_CR_BASE3	Cr base of index3
0x0000016C	RW	0x0	CMD_SET_FB_ADDR_FBC_Y_OFFSET3	FBC luma offset base of index3
0x00000170	RW	0x0	CMD_SET_FB_ADDR_FBC_C_OFFSET3	FBC chroma offset base of index3
0x00000174	RW	0x0	CMD_SET_FB_ADDR_LUMA_BASE4	Luma base of index4
0x00000178	RW	0x0	CMD_SET_FB_ADDR_CB_BASE4	Cb base of index4
0x0000017C	RW	0x0	CMD_SET_FB_ADDR_CR_BASE4	Cr base of index4
0x0000017C	RW	0x0	CMD_SET_FB_ADDR_FBC_Y_OFFSET4	FBC luma offset base of index4
0x00000180	RW	0x0	CMD_SET_FB_ADDR_FBC_C_OFFSET4	FBC chroma offset base of index4
0x00000184	RW	0x0	CMD_SET_FB_ADDR_LUMA_BASE5	Luma base of index5
0x00000188	RW	0x0	CMD_SET_FB_ADDR_CB_BASE5	Cb base of index5
0x0000018C	RW	0x0	CMD_SET_FB_ADDR_CR_BASE5	Cr base of index5

Offset	Type	Reset Value	Name	Description
0x0000018C	RW	0x0	*CMD_SET_FB_ADDR_FBC_Y_OFFSET5*	FBC luma offset base of index5
0x00000190	RW	0x0	*CMD_SET_FB_ADDR_FBC_C_OFFSET5*	FBC chroma offset base of index5
0x00000194	RW	0x0	*CMD_SET_FB_ADDR_LUMA_BASE6*	Luma base of index6
0x00000198	RW	0x0	*CMD_SET_FB_ADDR_CB_BASE6*	Cb base of index6
0x0000019C	RW	0x0	*CMD_SET_FB_ADDR_CR_BASE6*	Cr base of index6
0x0000019C	RW	0x0	*CMD_SET_FB_ADDR_FBC_Y_OFFSET6*	FBC luma offset base of index6
0x000001A0	RW	0x0	*CMD_SET_FB_ADDR_FBC_C_OFFSET6*	FBC chroma offset base of index6
0x000001A4	RW	0x0	*CMD_SET_FB_ADDR_LUMA_BASE7*	Luma base of index7
0x000001A8	RW	0x0	*CMD_SET_FB_ADDR_CB_BASE7*	Cb base of index7
0x000001AC	RW	0x0	*CMD_SET_FB_ADDR_CR_BASE7*	Cr base of index7
0x000001AC	RW	0x0	*CMD_SET_FB_ADDR_FBC_Y_OFFSET7*	FBC luma offset base of index7
0x000001B0	RW	0x0	*CMD_SET_FB_ADDR_FBC_C_OFFSET7*	FBC chroma offset base of index7
0x000001B4	RW	0x0	*CMD_SET_FB_ADDR_MV_COL0*	Colocated mv buffer base of index 0
0x000001B8	RW	0x0	*CMD_SET_FB_ADDR_MV_COL1*	Colocated mv buffer base of index 1
0x000001BC	RW	0x0	*CMD_SET_FB_ADDR_MV_COL2*	Colocated mv buffer base of index 2 (HEVC encoder only)
0x000001C0	RW	0x0	*CMD_SET_FB_ADDR_MV_COL3*	Colocated mv buffer base of index 3 (HEVC encoder only)
0x000001C4	RW	0x0	*CMD_SET_FB_ADDR_MV_COL4*	Colocated mv buffer base of index 4 (HEVC encoder only)
0x000001C8	RW	0x0	*CMD_SET_FB_ADDR_MV_COL5*	Colocated mv buffer base of index 5 (HEVC encoder only)
0x000001CC	RW	0x0	*CMD_SET_FB_ADDR_MV_COL6*	Colocated mv buffer base of index 6 (HEVC encoder only)
0x000001D0	RW	0x0	*CMD_SET_FB_ADDR_MV_COL7*	Colocated mv buffer base of index 7 (HEVC encoder only)
OUTPUT RETURN				

1.2.3.2.11. ENC_PIC Command Parameter Registers

Table 1.15. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000118	RW	0x0	*CMD_BS_START*	Bitstream buffer start address
0x0000011C	RW	0x0	*CMD_BS_SIZE*	Bitstream buffer size
0x00000120	RW	0x0	*CMD_BS_OPTIONS*	Bistream buffer option
0x00000124	RW	0x0	*USE_SEC_AXI*	Secondary AXI usage option
0x00000128	RW	0x0	*CMD_ENC_REPORT_PARAM*	Report parameters
0x0000012C	RW	0x0	*CMD_ENC_REPORT_ENDIAN*	Report buffer endianness
0x00000130	RW	0x0	*CMD_ENC_RESERVED*	Reserved
0x00000138	RW	0x0	*CMD_ENC_CUSTOM_MAP_OPTION_PARAM*	Custom map parameters
0x0000013C	RW	0x0	*CMD_ENC_CUSTOM_MAP_OPTION_ADDR*	Custom map address
0x00000144	RW	0x0	*CMD_ENC_SRC_PIC_IDX*	Buffer index of source picture
0x00000148	RW	0x0	*CMD_ENC_SRC_ADDR_Y*	Y component source address
0x0000014C	RW	0x0	*CMD_ENC_SRC_ADDR_U*	Cb component source address
0x00000150	RW	0x0	*CMD_ENC_SRC_ADDR_V*	Cr component source address

Offset	Type	Reset Value	Name	Description
0x00000154	RW	0x0	CMD_ENC_SRC_STRIDE	Stride of source picture
0x00000158	RW	0x0	CMD_ENC_SRC_FORMAT	Format of source picture
0x00000160	RW	0x0	CMD_ENC_SRC_AXI_SEL	Selection of source AXI port
0x00000164	RW	0x0	CMD_ENC_CODE_OPTION	NAL unit coding options
0x00000168	RW	0x0	CMD_ENC_PIC_PARAM	HEVC encoder picture level parameter
0x0000016C	RW	0x0	CMD_ENC_LONGTERM_PIC	Longterm picture setting
0x00000170	RW	0x0	CMD_ENC_WP_PIXEL_SIGMA_Y	Luma varience for weighted prediction
0x00000174	RW	0x0	CMD_ENC_WP_PIXEL_SIGMA_C	Cb varience for weighted prediction
0x00000178	RW	0x0	CMD_ENC_WP_PIXEL_MEAN_Y	Cr varience for weighted prediction
0x0000017C	RW	0x0	CMD_ENC_WP_PIXEL_MEAN_C	Pixel mean values for weighted prediction
0x00000180	RW	0x0	CMD_ENC_PIC_LF_PARAM_0	SVAC loopfilter parameters
0x00000184	RW	0x0	CMD_ENC_PIC_LF_PARAM_1	SVAC user filter enable flag and user filter level
0x00000188	RW	0x0	CMD_ENC_SRC_CF50_ADDR_Y_OFFSET	Base Address for Offset Table Y
0x0000018C	RW	0x0	CMD_ENC_SRC_CF50_ADDR_CB_OFFSET	Base Address for Offset Table Cb
0x00000190	RW	0x0	CMD_ENC_SRC_CF50_ADDR_CR_OFFSET	Base Address for Offset Table Cr
OUTPUT RETURN				

1.2.3.2.12. QUERY Command Parameter Registers

1.2.3.2.12.1. GET_VPU_INFO

Table 1.16. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000104	RW	0x0	CMD_QUERY_OPTION	QUERY command option
OUTPUT RETURN				
0x00000118	RW	0x0	RET_QUERY_FW_VERSION	Firmware Version
0x0000011C	RW	0x0	RET_QUERY_PRODUCT_NAME	HW product name
0x00000120	RW	0x0	RET_QUERY_PRODUCT_VERSION	HW product version
0x00000124	RW	0x0	RET_QUERY_STD_DEF0	Standard definition
0x00000128	RW	0x0	RET_QUERY_STD_DEF1	Standard definition
0x0000012C	RW	0x0	RET_QUERY_CONF_FEATURE	Configuration feature
0x00000130	RW	0x0	RET_QUERY_CONF_DATE	Configuration date
0x00000134	RW	0x0	RET_QUERY_CONF_REVISION	The revision of H/W configuration
0x00000138	RW	0x0	RET_QUERY_CONF_TYPE	The define value of H/W configuration
0x0000013C	RW	0x0	RET_QUERY_PRODUCT_ID	The product ID
0x00000140	RW	0x0	RET_QUERY_CUSTOMER_ID	The customer ID

1.2.3.2.12.2. GET_RESULT for Encoder

Table 1.17. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000104	RW	0x0	CMD_QUERY_OPTION	QUERY command option
OUTPUT RETURN				

Offset	Type	Reset Value	Name	Description
0x00000114	RW	0x0	*RET_QUERY_ENC_RD_PTR*	Bistream buffer read pointer
0x00000118	RW	0x0	*RET_QUERY_ENC_WR_PTR*	Bistream buffer write pointer
0x0000011C	RW	0x0	*RET_QUERY_ENC_NUM_REQUIRED_FB*	Minimum number of reference frame buffer required for encoding
0x00000120	RW	0x0	*RET_QUERY_MIN_SRC_BUF_NUM*	Minimum number of source frame buffer required for encoding
0x00000124	RW	0x0	*RET_QUERY_ENC_PIC_TYPE*	Encoded picture type
0x00000128	RW	0x0	*RET_QUERY_ENC_PIC_POC*	A POC value of encoded picture
0x0000012C	RW	0x0	*RET_QUERY_ENC_PIC_IDX*	Frame buffer index of encoded picture
0x00000130	RW	0x0	*RET_QUERY_ENC_PIC_SLICE_NUM*	Number of slice segments
0x00000134	RW	0x0	*RET_QUERY_ENC_PIC_SKIP*	A picture skip flag
0x00000138	RW	0x0	*RET_QUERY_ENC_PIC_NUM_INTRA*	Number of intra block
0x0000013C	RW	0x0	*RET_QUERY_ENC_PIC_NUM_MERGE*	Number of merge block
0x00000140	RW	0x0	*RET_QUERY_ENC_PIC_FLAG*	RESERVED
0x00000144	RW	0x0	*RET_QUERY_ENC_PIC_NUM_SKIP*	Number of skip block
0x00000148	RW	0x0	*RET_QUERY_ENC_PIC_AVG_CTU_QP*	CTU QP on average
0x0000014C	RW	0x0	*RET_QUERY_ENC_PIC_BYTE*	Byte size of encoded picture
0x00000150	RW	0x0	*RET_QUERY_ENC_GOP_PIC_IDX*	A picture index in GOP
0x00000154	RW	0x0	*RET_QUERY_ENC_USED_SRC_IDX*	Buffer index of source picture that is used for encoding
0x00000158	RW	0x0	*RET_QUERY_ENC_PIC_NUM*	Encoded picture number
0x0000015C	RW	0x0	*RET_QUERY_ENC_NUT_0*	Encoded NAL unit type of VCL
0x00000160	RW	0x0	*RET_QUERY_ENC_NUT_1*	Encoded NAL unit type of VCL
0x00000164	RW	0x0	*RET_QUERY_ENC_PIC_DIST_LOW*	Low 32bit SSD
0x00000168	RW	0x0	*RET_QUERY_ENC_PIC_DIST_HIGH*	High 32bit SSD
0x0000016C	RW	0x0	*RET_QUERY_ENC_MAX_LATENCY_PICTURES*	The number of latency picture
0x00000170	RW	0x0	*RET_QUERY_ENC_SVC_LAYER*	SVC layer type of the encoded picture
0x00000178	RW	0x0	*RET_QUERY_REPORT_PARAM*	Report parameters
0x0000017C	RW	0x0	*RET_QUERY_ENC_REPORT_BASE*	Report buffer base address
0x000001B8	RW	0x0	*RET_QUERY_HOST_CMD_TICK*	Host cmd queue tick
0x000001BC	RW	0x0	*RET_QUERY_DEC_PREPARE_START_TICK*	Prepare start tick
0x000001C0	RW	0x0	*RET_QUERY_DEC_PREPARE_END_TICK*	Preprae end tick
0x000001C4	RW	0x0	*RET_QUERY_DEC_PROCESSING_START_TICK*	Processing start tick
0x000001C8	RW	0x0	*RET_QUERY_DEC_PROCESSING_END_TICK*	Processing end tick
0x000001CC	RW	0x0	*RET_QUERY_DEC_ENCODING_START_TICK*	Encoding start tick
0x000001D0	RW	0x0	*RET_QUERY_DEC_ENCODING_END_TICK*	Encoding end tick
0x000001D8	RW	0x0	*RET_QUERY_ENC_ERR_INFO*	Error information
0x000001DC	RW	0x0	*RET_QUERY_ENC_SUCCESS*	Query result

1.2.3.2.12.3. GET_BS_WR_PTR

Table 1.18. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000104	RW	0x0	*CMD_QUERY_OPTION*	QUERY command option
0x0000011C	RW	0x0	*CMD_QUERY_ENC_REASON_SEL*	The report type of bitstream buffer position

Offset	Type	Reset Value	Name	Description
OUTPUT RETURN				
0x00000114	RW	0x0	*RET_QUERY_ENC_BS_RD_PTR*	The start position of bitstream buffer
0x00000118	RW	0x0	*RET_QUERY_ENC_BS_WR_PTR*	The end position of bitstream buffer

1.2.3.2.13. UPDATE_BS Parameter Registers for Encoder

Table 1.19. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000104	RW	0x0	*CMD_UPDATE_BS_OPTION*	Run command option
0x00000118	RW	0x0	*CMD_BS_START*	Bitstream buffer start address
0x0000011C	RW	0x0	*CMD_BS_SIZE*	Bitstream buffer size
0x00000120	RW	0x0	*CMD_BS_OPTIONS*	Bistream buffer option
OUTPUT RETURN				

1.2.4. Register Descriptions

Detailed definitions of host interface registers are presented in the following sub-sections. It covers general description, bit assignment, and meaning of bit field of Command I/O registers.

1.2.4.1. Control Register Descriptions

1.2.4.1.1. VPU_PO_CONF (0x00000000)

Power On Configurations

Table 1.20. VPU_PO_CONF Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																												USE_PO_CONF	RSVD		DEBUGMODE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	WO	R	R	WO

Table 1.21. VPU_PO_CONF Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3]	USE_PO_CONF	WO	USE PO_CONF Host processor should set *0* for this field which is required when VPU initialization.	0x0
[2:1]	RSVD	R	Reserved	0x0
[0]	DEBUGMODE	WO	Power on with debug mode Host processor should set *0* for this field.	0x0

1.2.4.1.2. VCPU_CUR_PC (0x00000004)

Current program counter value of V-CPU

Table 1.22. VCPU_CUR_PC Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUR_PC																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 1.23. VCPU_CUR_PC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CUR_PC	RO	PC value represents the address of instruction which is executed in V-CPU. (for debugging purpose only)	0x0

1.2.4.1.3. VCPU_CUR_LR (0x00000008)

Current LR (for debugger only)

Table 1.24. VCPU_CUR_LR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUR_LR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 1.25. VCPU_CUR_LR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CUR_LR	RO	Current LR (Link Register) to find out caller address	0x0

1.2.4.1.4. VPU_PDBG_STEP_MASK (0x0000000C)

Debugger control
(for debugger only)

Table 1.26. VPU_PDBG_STEP_MASK Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															STEP_MASK_ENABLE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Table 1.27. VPU_PDBG_STEP_MASK Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	STEP_MASK_ENABLE	RW	Interrupt Disable at step for debugger	0x0

1.2.4.1.5. VPU_PDBG_CTRL (0x00000010)

Debugger control
(for debugger only)

Table 1.28. VPU_PDBG_CTRL Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																												IMMBRK	STABLEBRK	RESUME	STEP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	WO	WO	WO	WO

Table 1.29. VPU_PDBG_CTRL Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[3]	IMMBRK	WO	Immediate break It forces to stop V-CPU and enter in a debug mode to analysis hang-up situation. V-CPU cannot resume from the break point.	0x0
[2]	STABLEBRK	WO	Stable break It is an external break request when V-CPU is in stable (breakable) status.	0x0
[1]	RESUME	WO	Resume It resumes from the breakpoint.	0x0
[0]	STEP	WO	Step It runs V-CPU in step instruction mode.	0x0

1.2.4.1.6. VPU_PDBG_IDX_REG (0x00000014)

V-CPU debugger index register
(For debugger only)

Table 1.30. VPU_PDBG_IDX_REG Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						RDDBG	WRDBG	DBGIDX							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Table 1.31. VPU_PDBG_IDX_REG Field Description

Bit	Name	Type	Function	Reset Value
[31:10]	RSVD	R	Reserved	0x0
[9]	RDDBG	WO	Read Operation Request RDDBG and WRDBG cannot be 1 at the same time.	0x0
[8]	WRDBG	WO	Write Operation Request RDDBG and WRDBG cannot be 1 at the same time.	0x0
[7:0]	DBGIDX	WO	Debug Index Debugger Register Index	0x0

1.2.4.1.7. VPU_PDBG_WDATA_REG (0x00000018)

V-CPU debugger write data register
(For debugger only)

Table 1.32. VPU_PDBG_WDATA_REG Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPU_PDBG_WDATA_REG																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo

Table 1.33. VPU_PDBG_WDATA_REG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	VPU_PDBG_WDATA_REG	WO	To write data to the debugger, 1. Write some data to this register. 2. Write VCPU_PDBG_IDX_REG with WRDBG as 1 and DBGIDX as this register address. 3. After writing is completed, VPU_PDBG_WDATA_REG will be cleared.	0x0

1.2.4.1.8. VPU_PDBG_RDATA_REG (0x0000001C)

V-CPU debugger read data register
(For debugger only)

Table 1.34. VPU_PDBG_RDATA_REG Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPU_PDBG_RDATA_REG																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 1.35. VPU_PDBG_RDATA_REG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	VPU_PDBG_RDATA_REG	RO	To read data to the debugger, 1. Write VCPU_PDBG_IDX_REG with RDDBG as 1 and DBGIDX as the register address to be read. 2. Read from the specified register to this register.	0x0

1.2.4.1.9. VPU_FIO_CTRL_ADDR (0x00000020)

FIO CRTL, READY, and Address for accessing FIO (FastIO) which is an internal peripheral bus in VPU.
By accessing FIO, user can check the internal status of some accelerators in VPU for debugging purpose in some cases.
FIO transaction is carried out when host writes this register.
CAUTON> Because accessing FIO can make unwanted behavior in VPU, please access it using API only.

Table 1.36. VPU_FIO_CTRL_ADDR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READY	RSVD														RW_FLAG	FIO_ADDR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO

Table 1.37. VPU_FIO_CTRL_ADDR Field Description

Bit	Name	Type	Function	Reset Value
[31]	READY	RW	Ready for the transaction When writing, it should be 0.	0x0
[30:17]	RSVD	R	Reserved	0x0
[16]	RW_FLAG	WO	Read/Write transaction control 0: read 1: write	0x0
[15:0]	FIO_ADDR	WO	FIO Address	0x0

1.2.4.1.10. VPU_FIO_DATA (0x00000024)

FIO data

Table 1.38. VPU_FIO_DATA Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIO_DATA																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.39. VPU_FIO_DATA Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FIO_DATA	RW	When writing, FIO data should be written to this register firstly. When reading, 1. Write data to this register. 2. Check whether READY flag of VPU_FIO_CTRL_ADDR register is 1. 3. When READY flag is asserted, VPU reads data from this VPU_FIO_DATA.	0x0

1.2.4.1.11. VPU_VINT_REASON_USR (0x00000030)

Interrupt Reason Check & Clear using user level previlege in the host

- When an interrupt is asserted, the value of VPU_VINT_REASON is ORing to VPU_VINT_REASON_USER.
- Even if interrupt service routine(ISR) clears VPU_INT_REASON by using VPU_VINT_REASON_CLR, an application in user mode can identify the reason of the interrupt signalling from VPU by accessing this register.
- Applications in user mode should clear this register just after perform processing for the interrupt.

Table 1.40. VPU_VINT_REASON_USR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																BSEMPTY_INTR_USER	CMDE_INTR_USER	CMDD_INTR_USER	RSVD	RSVD	RSVD	CMD9_INTR_USER	CMD8_INTR_USER	CMD7_INTR_USER	CMD6_INTR_USER	CMD5_INTR_USER	CMD4_INTR_USER	CMD3_INTR_USER	CMD2_INTR_USER	CMD1_INTR_USER	CMD0_INTR_USER

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.41. VPU_VINT_REASON_USR Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15]	BSEMPTY_INTR_USER	RW	Bitstream empty(feeding request) interrupt	0x0
[14]	CMDE_INTR_USER	RW	QUERY command done interrupt	0x0
[13]	CMDD_INTR_USER	RW	Low latency interrupt	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	CMD9_INTR_USER	RW	ENC_SET_PARAM command done interrupt	0x0
[8]	CMD8_INTR_USER	RW	DEC_PIC/ENC_PIC command done interrupt	0x0
[7]	CMD7_INTR_USER	RW	SET_FRAMEBUFFER command done interrupt	0x0
[6]	CMD6_INTR_USER	RW	INIT_SEQ command done interrupt	0x0
[5]	CMD5_INTR_USER	RW	DESTROY_INSTANCE command done interrupt	0x0
[4]	CMD4_INTR_USER	RW	FLSUH_INSTANCE command done interrupt	0x0
[3]	CMD3_INTR_USER	RW	CREATE_INSTANCE command done interrupt	0x0
[2]	CMD2_INTR_USER	RW	SLEEP_VPU command done interrupt	0x0
[1]	CMD1_INTR_USER	RW	WAKE_VPU command done interrupt	0x0
[0]	CMD0_INTR_USER	RW	INIT_VPU command done interrupt	0x0

1.2.4.1.12. VPU_VINT_REASON_CLR (0x00000034)

Interrupt Reason Clear
This register clears the interrupt reason in ISR when host processor catchs an interrupt. It is to notify that host processor has received the interrupt. If host processor wants to get task done interrupts or so from VPU, they should set the fields of VPU_VINT_ENABLE register as they wish. And when they recieve any of the command done interrupt from VPU (VPU_VINT_REASON_CLR is not zero), host processor should check the interrupt and do the following sequence for safe interrupt clear.

1. Clear the interrupt by setting the relevant bit of this VPU_VINT_REASON_CLR.
2. Set VPU_VINT_CLEAR to 1, which drops the interrupt signal.
3. Then VPU_VPU_INT_STS is automatically cleared.

In above sequence, 1 is important because VPU_VINT_CLEAR without VPU_VINT_REASON_CLR might lead to interrupt reassurtion. It was because interrupts happened concurrently and other interrupt is still remaining even though one was cleared.

Table 1.42. VPU_VINT_REASON_CLR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																BSEMPTY_CLR	CMDE_CLR	CMDD_CLR	RSVD	RSVD	RSVD	CMD9_CLR	CMD8_CLR	CMD7_CLR	CMD6_CLR	CMD5_CLR	CMD4_CLR	CMD3_CLR	CMD2_CLR	CMD1_CLR	CMD0_CLR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.43. VPU_VINT_REASON_CLR Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15]	BSEMPTY_CLR	RW	Bitstream empty (bitstream feeding request) interrupt clear	0x0
[14]	CMDE_CLR	RW	QUERY command done interrupt clear	0x0
[13]	CMDD_CLR	RW	Low latency interrupt clear	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	CMD9_CLR	RW	ENC_SET_PARAM command done interrupt clear	0x0
[8]	CMD8_CLR	RW	DEC_PIC/ENC_PIC command done interrupt clear	0x0
[7]	CMD7_CLR	RW	SET_FRAMEBUFFER command done interrupt clear	0x0
[6]	CMD6_CLR	RW	INIT_SEQ command done interrupt clear	0x0
[5]	CMD5_CLR	RW	DESTROY_INSTANCE command done interrupt clear	0x0
[4]	CMD4_CLR	RW	FLSUH_INSTANCE command done interrupt clear	0x0
[3]	CMD3_CLR	RW	CREATE_INSTANCE command done interrupt clear	0x0
[2]	CMD2_CLR	RW	SLEEP_VPU command done interrupt clear	0x0
[1]	CMD1_CLR	RW	WAKE_VPU command done interrupt clear	0x0
[0]	CMD0_CLR	RW	INIT_VPU command done interrupt clear	0x0

1.2.4.1.13. VPU_HOST_INT_REQ (0x00000038)

Interrupt request sent from host processor to VPU for the command and so on.

Table 1.44. VPU_HOST_INT_REQ Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															HINTREQ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Table 1.45. VPU_HOST_INT_REQ Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	HINTREQ	RW	If this is set to 1, an interrupt named HOST interrupt is sent to VPU.	0x0

1.2.4.1.14. VPU_VINT_CLEAR (0x0000003C)

VPU interrupt clear

Table 1.46. VPU_VINT_CLEAR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															VINTCLR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Table 1.47. VPU_VINT_CLEAR Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	VINTCLR	RW	Host processor can clear the VPU interrupt which has been pending through this register. As mentioned in VPU_VINT_REASON_CLR, this register setting definetely come after VPU_INT_REASON setting. Also, setting this VPU_VINT_CLEAR to 1 forces VPU_VPU_INT_STS INT_STS to be cleared automatically.	0x0

1.2.4.1.15. VPU_HINT_CLEAR (0x00000040)

Host interrupt clear

Table 1.48. VPU_HINT_CLEAR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															HINTCLR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Table 1.49. VPU_HINT_CLEAR Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	HINTCLR	RW	VPU can clear the host interrupt which has been pending through this register. When VPU operation for the host interrupt is done, VPU sets this reigster to clear the host interrupt.	0x0

1.2.4.1.16. VPU_VPU_INT_STS (0x00000044)

Interrupt Status

Table 1.50. VPU_VPU_INT_STS Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															VPU_VPU_INT_STS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Table 1.51. VPU_VPU_INT_STS Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	VPU_VPU_INT_STS	RW	VPU Interrupt status which comes from VPU to Host This register turns 1 if VPU_VINT_REASON is not zero (any bit of VPU_VINT_REASON is on), and it becomes to be clear by setting VPU_VINT_CLEAR register. This is a way of interrupt check by register polling.	0x0

1.2.4.1.17. VPU_VINT_ENABLE (0x00000048)

Interrupt Enable for each interrupt reason(source)
Interrupt in LSB position shall be handled with higher priority.

Table 1.52. VPU_VINT_ENABLE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																CMDF_EN	CMDE_EN	CMDD_EN	RSVD	RSVD	RSVD	CMD9_EN	CMD8_EN	CMD7_EN	CMD6_EN	CMD5_EN	CMD4_EN	CMD3_EN	CMD2_EN	CMD1_EN	CMD0_EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.53. VPU_VINT_ENABLE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15]	CMDF_EN	RW	UPDATE_BS command done interrupt enable	0x0
[14]	CMDE_EN	RW	QUERY command done interrupt enable	0x0
[13]	CMDD_EN	RW	Low latency interrupt enable	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	CMD9_EN	RW	ENC_SET_PARAM command done interrupt enable	0x0
[8]	CMD8_EN	RW	DEC_PIC/ENC_PIC command done interrupt enable	0x0
[7]	CMD7_EN	RW	SET_FRAMEBUFFER command done interrupt enable	0x0
[6]	CMD6_EN	RW	INIT_SEQ command done interrupt enable	0x0
[5]	CMD5_EN	RW	DESTROY_INSTANCE command done interrupt enable	0x0
[4]	CMD4_EN	RW	FLSUH_INSTANCE command done interrupt enable	0x0
[3]	CMD3_EN	RW	CREATE_INSTANCE command done interrupt enable	0x0
[2]	CMD2_EN	RW	SLEEP_VPU command done interrupt enable	0x0
[1]	CMD1_EN	RW	WAKE_VPU command done interrupt enable	0x0
[0]	CMD0_EN	RW	INIT_VPU command done interrupt enable	0x0

1.2.4.1.18. VPU_VINT_REASON (0x0000004C)

VPU interrupt reason
This register Interrupt reasons are almost the same with the command. Interrupt in LSB position shall be handled with higher priority.

Table 1.54. VPU_VINT_REASON Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																BSEMPTY_INTR	CMDE_INTR	CMDD_INTR	RSVD	RSVD	RSVD	CMD9_INTR	CMD8_INTR	CMD7_INTR	CMD6_INTR	CMD5_INTR	CMD4_INTR	CMD3_INTR	CMD2_INTR	CMD1_INTR	CMD0_INTR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.55. VPU_VINT_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15]	BSEMPTY_INTR	RW	Bitstream empty (bitstream feeding request)	0x0
[14]	CMDE_INTR	RW	QUERY command done interrupt	0x0
[13]	CMDD_INTR	RW	Low latency interrupt	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	CMD9_INTR	RW	ENC_SET_PARAM command done interrupt	0x0
[8]	CMD8_INTR	RW	DEC_PIC/ENC_PIC command done interrupt	0x0
[7]	CMD7_INTR	RW	SET_FRAMEBUFFER command done interrupt	0x0
[6]	CMD6_INTR	RW	INIT_SEQ command done interrupt	0x0
[5]	CMD5_INTR	RW	DESTROY_INSTANCE command done interrupt	0x0
[4]	CMD4_INTR	RW	FLSUH_INSTANCE command done interrupt	0x0
[3]	CMD3_INTR	RW	CREATE_INSTANCE command done interrupt	0x0
[2]	CMD2_INTR	RW	SLEEP_VPU command done interrupt	0x0
[1]	CMD1_INTR	RW	WAKE_VPU command done interrupt	0x0
[0]	CMD0_INTR	RW	INIT_VPU command done interrupt	0x0

1.2.4.1.19. VPU_RESET_REQ (0x00000050)

VPU reset request for each block of clock domain
0: reset request clear or do noting
1: reset request For more details, please refer to *clock and reset signals* section in datasheet.

Table 1.56. VPU_RESET_REQ Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					VCRST_REQ	VBRST_REQ	VARST_REQ	ARST_REQ								BRST_REQ								CRST_REQ							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.57. VPU_RESET_REQ Field Description

Bit	Name	Type	Function	Reset Value
[31:27]	RSVD	R	Reserved	0x0
[26]	VCRST_REQ	RW	CCLK domain (for V-CPU) reset request	0x0
[25]	VBRST_REQ	RW	BCLK domain (for V-CPU) reset request	0x0
[24]	VARST_REQ	RW	ACLK domain (for V-CPU) reset request	0x0
[23:16]	ARST_REQ	RW	ACLK domain (for each vCORE) reset request [23-20] Reserved [19] : V-CORE3 [18] : V-CORE2 [17] : V-CORE1 [16] : V-CORE0	0x0

Bit	Name	Type	Function	Reset Value
[15:8]	BRST_REQ	RW	BCLK domain (for each vCORE) reset request [15:12]: Reserved [11] : V-CORE3 [10] : V-CORE2 [9] : V-CORE1 [8] : V-CORE0	0x0
[7:0]	CRST_REQ	RW	CCLK domain (for each vCORE) reset request [7:4]: Reserved [3] : V-CORE3 [2] : V-CORE2 [1] : V-CORE1 [0] : V-CORE0	0x0

1.2.4.1.20. VPU_RESET_STATUS (0x00000054)

Reset status for each clock domain
0: reset is released
1: on reset handling phase

Table 1.58. VPU_RESET_STATUS Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					VCRST_STS	VBRST_STS	VARST_STS	ARST_STS								BRST_STS								CRST_STS							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 1.59. VPU_RESET_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:27]	RSVD	R	Reserved	0x0
[26]	VCRST_STS	RO	CCLK domain (for V-CPU) reset status	0x0
[25]	VBRST_STS	RO	BCLK domain (for V-CPU) reset status	0x0
[24]	VARST_STS	RO	ACLK domain (for V-CPU) reset status	0x0
[23:16]	ARST_STS	RO	ACLK domain (for each vCORE) reset status	0x0
[15:8]	BRST_STS	RO	BCLK domain (for each vCORE) reset status	0x0
[7:0]	CRST_STS	RO	CCLK domain (for each vCORE) reset status	0x0

1.2.4.1.21. VCPU_RESTART (0x00000058)

V-CPU restart request

Table 1.60. VCPU_RESTART Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															VCPU_RESTART
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	WO

Table 1.61. VCPU_RESTART Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	VCPU_RESTART	WO	This register restarts V-CPU from the reset vector without clearing H/W logic. 0: DO NOTHING 1: RESTART REQUEST	0x0

1.2.4.1.22. VPU_CLK_MASK (0x0000005C)

Clock gating control register for each clock domain

Table 1.62. VPU_CLK_MASK Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					CCLK_CPU_EN	VCLK_CPU_EN	ACLK_CPU_EN	ACLK_EN								BCLK_EN								CCLK_EN							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.63. VPU_CLK_MASK Field Description

Bit	Name	Type	Function	Reset Value
[31:27]	RSVD	R	Reserved	0x0
[26]	CCLK_CPU_EN	RW	CCLK domain (for V-CPU) gating	0x0
[25]	VCLK_CPU_EN	RW	BCLK domain (for V-CPU) gating	0x0
[24]	ACLK_CPU_EN	RW	ACLK domain (for V-CPU) gating	0x0
[23:16]	ACLK_EN	RW	ACLK domain (for each V-CORE) gating	0x0
[15:8]	BCLK_EN	RW	BCLK domain (for each V-CORE) gating	0x0
[7:0]	CCLK_EN	RW	CCLK domain (for each V-CORE) gating	0x0

1.2.4.1.23. VPU_REMAP_CTRL (0x00000060)

Remap control register (REMAP REG ADDR / ATTR / SIZE)
VPU remaps addresses it uses between physical memory and virtual memory for efficient address management. VPU works with virtual memory addresses that are translated to phsyical addresses. Host processor needs to assign V-CPU code buffer to VPU_REMAP_PADDR and VPU_REMAP_VADDR to 0, so that VPU can start by reading from VPU_REMAP_PADDR considering it is its base address 0. This register has configuration fields for remapping.

Table 1.64. VPU_REMAP_CTRL Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REMAP_GLOBEN	RSVD							AXIID_PROC				ENDIAN				REMAP_IDX				REMAP_PAGE_SIZE_EN	RSVD			REMAP_PSIZE							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.65. VPU_REMAP_CTRL Field Description

Bit	Name	Type	Function	Reset Value
[31]	REMAP_GLOBEN	RW	Global Control Update Enable [30:16] is valid only if this flag is *1*. Thus, global configurations such as AXIID_PROC and ENDIAN is valid only if REMAP_GLOBEN is *1*. If not, the value in the fields are ignored.	0x0
[30:24]	RSVD	RW	Reserved	0x0
[23:20]	AXIID_PROC	RW	Upper AXI-ID for processor bus to distinguish guest OS For virtualization only. Use this value from higher bit of the 4 bits.	0x0
[19:16]	ENDIAN	RW	Endianness for memory access Endian control of memory transactions from processor core	0x0
[15:12]	REMAP_IDX	RW	Remap index Only 0 to 3 are valid.	0x0
[11]	REMAP_PAGE_SIZE_EN	RW	Enable to update a Remap Page Size. In other words, Remap Page Size can change only if this bit is 1.	0x0
[10:9]	RSVD	RW	Reserved	0x0
[8:0]	REMAP_PSIZE	RW	Remap Page Size (page base should be aligned in 4K region) 0x001: 4K 0x002: 8K 0x004: 16K 0x008: 32K 0x010 : 64K 0x020 : 128K 0x040 : 256K 0x080: 512K 0x100: 1M	0x0

1.2.4.1.24. VPU_REMAP_VADDR (0x00000064)

Remap region base address in virtual address space

Table 1.66. VPU_REMAP_VADDR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPU_REMAP_VADDR																				RSVD											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.67. VPU_REMAP_VADDR Field Description

Bit	Name	Type	Function	Reset Value
[31:12]	VPU_REMAP_VADDR	RW	Virtual address space is an address generated by V-CPU.	0x0

Bit	Name	Type	Function	Reset Value
			Section address should be aligned to 4KB boundary. CAUTION> In case of CODE section (which has REMAP IDX 0), virtual address for the section should be 0x0, since V-CPU always start booting up from virtual address 0. If not, VPU can not boot-up.	
[11:0]	RSVD	R	Reserved	0x0

1.2.4.1.25. VPU_REMAP_PADDR (0x00000068)

Remap region base address in physical address space

Table 1.68. VPU_REMAP_PADDR Bit Assignment

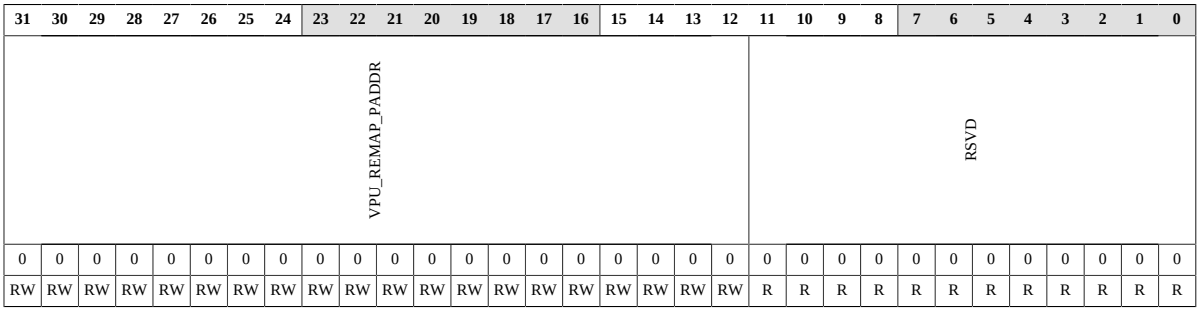


Table 1.69. VPU_REMAP_PADDR Field Description

Bit	Name	Type	Function	Reset Value
[31:12]	VPU_REMAP_PADDR	RW	Real address(physical address) as a pair of virtual address. It should aligned to 4KB boundary.	0x0
[11:0]	RSVD	R	Reserved	0x0

1.2.4.1.26. VPU_REMAP_CORE_START (0x0000006C)

VPU Start Request

Table 1.70. VPU_REMAP_CORE_START Bit Assignment

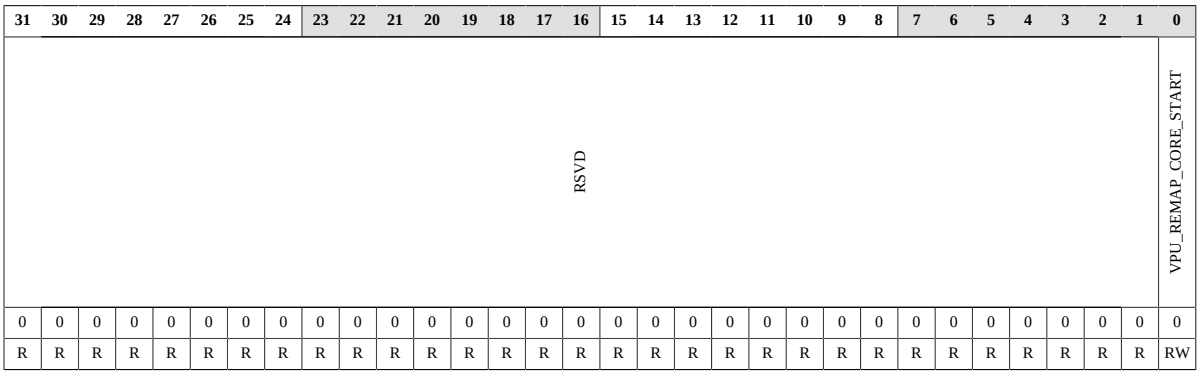


Table 1.71. VPU_REMAP_CORE_START Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	VPU_REMAP_CORE_START	RW	It starts VPU after initial setting has been done.	0x0

Bit	Name	Type	Function	Reset Value
			After setting up remap or initial configuartion, host should set this register to init VPU.	

1.2.4.1.27. VPU_BUSY_STATUS (0x00000070)

VPU Busy Status

Table 1.72. VPU_BUSY_STATUS Bit Assignment

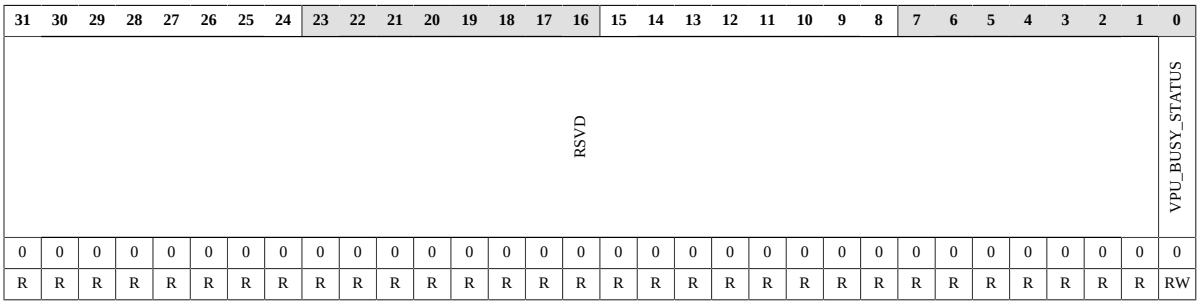


Table 1.73. VPU_BUSY_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	VPU_BUSY_STATUS	RW	Command Reentrance Check [0] - host should set this flag just before trying to send a command into command-queue - firmware clears this flag when a command is queueued for processing. - for the case of clock_gating or power_down, please use VPU_VCPU_STATUS register to check the status of VPU.	0x0

1.2.4.1.28. VPU_HALT_STATUS (0x00000074)

For power control, status checking of V-CPU

Table 1.74. VPU_HALT_STATUS Bit Assignment

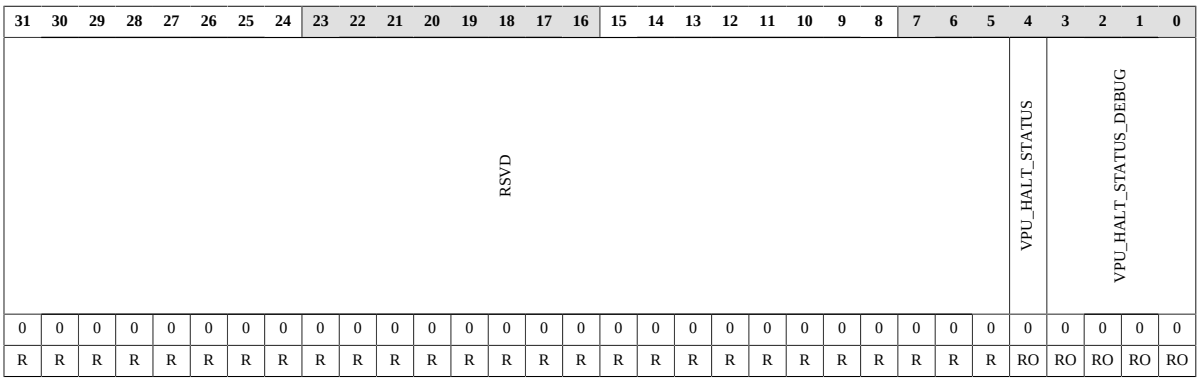


Table 1.75. VPU_HALT_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4]	VPU_HALT_STATUS	RO	1: V-CPU is on the HALT status	0x0

Bit	Name	Type	Function	Reset Value
[3:0]	VPU_HALT_STATUS_DEBUG	RO	for debugging - Eventhough V-CPU is on HALT sta- tus, for the clock gating or power down, VPU_VCPU_STATUS should be checked to check pending bus operations.	0x0

1.2.4.1.29. VPU_VCPU_STATUS (0x00000078)

V-CPU bus busy and V-CPU status

Table 1.76. VPU_VCPU_STATUS Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																VPU_VCPU_STATUS															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 1.77. VPU_VCPU_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:15]	RSVD	R	Reserved	0x0
[14:0]	VPU_VCPU_STATUS	RO	If [31:16] is 0x0000, there is no more bus transaction on V-CPU. If [15:0] is 0x0040, V-CPU is on the halt status. Thus, the value returns 0x40, power for VPU can be turnned-off	0x0

1.2.4.1.30. RSVD (0x0000007C)

Table 1.78. RSVD Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 1.79. RSVD Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	RO	Reserved	0x0

1.2.4.1.31. RET_FIO_STATUS (0x00000080)

Table 1.80. RET_FIO_STATUS Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Table 1.81. RET_FIO_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	RO	Reserved	0x0

1.2.4.1.32. RET_PRODUCT_NAME (0x00000090)

Table 1.82. RET_PRODUCT_NAME Bit Assignment

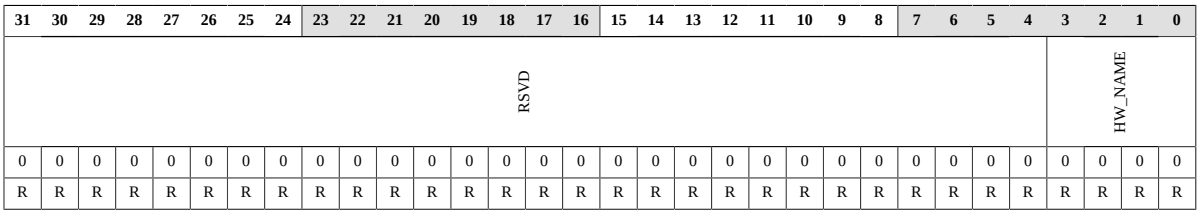


Table 1.83. RET_PRODUCT_NAME Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	HW_NAME	R	VPU hardware product name It always returns "WAVE" for WAVE5 series IP product.	0x0

1.2.4.1.33. RET_PRODUCT_VERSION (0x00000094)

Table 1.84. RET_PRODUCT_VERSION Bit Assignment

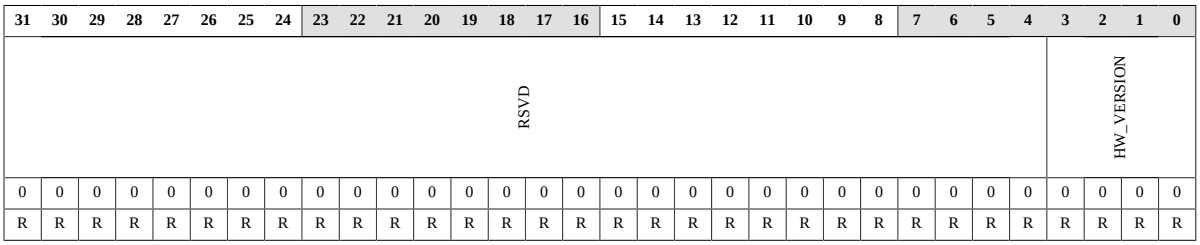
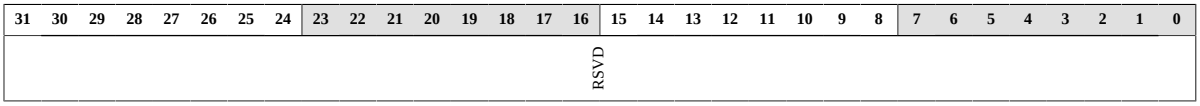


Table 1.85. RET_PRODUCT_VERSION Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	HW_VERSION	R	VPU hardware product version It returns as follows: 0x5120 for WAVE512 0x5150 for WAVE515 0x5110 for WAVE511 0x5200 for WAVE520 0x5250 for WAVE525 0x5210 for WAVE521	0x0

1.2.4.1.34. RET_VCPU_CONFIG0 (0x00000098)

Table 1.86. RET_VCPU_CONFIG0 Bit Assignment



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.87. RET_VCPU_CONFIG0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R	Configuration Information #0 (internal use only)	0x0

1.2.4.1.35. RET_VCPU_CONFIG1 (0x0000009C)

Table 1.88. RET_VCPU_CONFIG1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.89. RET_VCPU_CONFIG1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R	Configuration Information #1 (internal use only)	0x0

1.2.4.1.36. RET_CODEC_STD (0x000000A0)

Table 1.90. RET_CODEC_STD Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODEC_STD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.91. RET_CODEC_STD Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CODEC_STD	R	General Configuration Information (internal use only)	0x0

1.2.4.1.37. RET_CONF_DATE (0x000000A4)

Table 1.92. RET_CONF_DATE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW_DATE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.93. RET_CONF_DATE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	HW_DATE	R	The date that the hardware has been configured in YYYYmmdd in digit (internal use only)	0x0

1.2.4.1.38. RET_CONF_REVISION (0x000000A8)

Table 1.94. RET_CONF_REVISION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW_REVISION																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.95. RET_CONF_REVISION Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	HW_REVISION	R	The revision number when the hardware has been configured (internal use only)	0x0

1.2.4.1.39. RET_CONF_TYPE (0x000000AC)

Table 1.96. RET_CONF_TYPE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW_TYPE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.97. RET_CONF_TYPE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	HW_TYPE	R	The define value used in hardware configuration (internal use only)	0x0

1.2.4.1.40. RET_VCORE0_CFG (0x000000B0)

Table 1.98. RET_VCORE0_CFG Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONFIG_VORE0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.99. RET_VCORE0_CFG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CONFIG_VORE0	R	The VCORE0 Configuration Information (internal use only)	0x0

1.2.4.1.41. RET_VCORE1_CFG (0x000000B4)

Table 1.100. RET_VCORE1_CFG Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

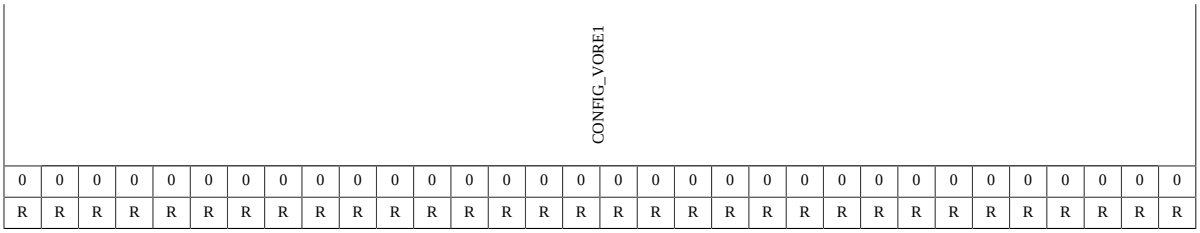


Table 1.101. RET_VCORE1_CFG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CONFIG_VORE1	R	The VCORE1 Configuration Information (internal use only)	0x0

1.2.4.1.42. RET_VCORE2_CFG (0x000000B8)

Table 1.102. RET_VCORE2_CFG Bit Assignment

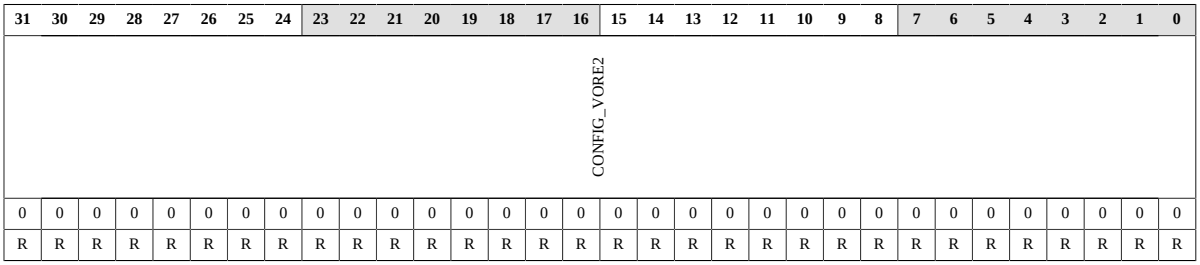


Table 1.103. RET_VCORE2_CFG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CONFIG_VORE2	R	The VCORE2 Configuration Information (internal use only)	0x0

1.2.4.1.43. RET_VCORE3_CFG (0x000000BC)

Table 1.104. RET_VCORE3_CFG Bit Assignment

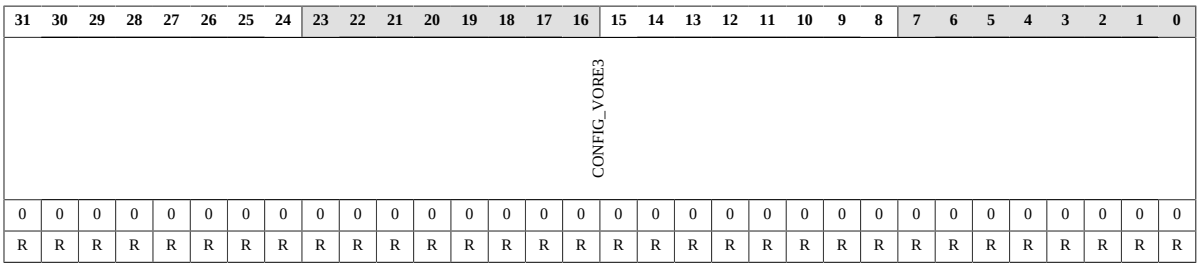
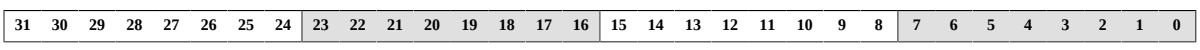


Table 1.105. RET_VCORE3_CFG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CONFIG_VORE3	R	The VCORE3 Configuration Information (internal use only)	0x0

1.2.4.1.44. VPU_RET_VCORE_PRESET (0x000000D0)

Table 1.106. VPU_RET_VCORE_PRESET Bit Assignment



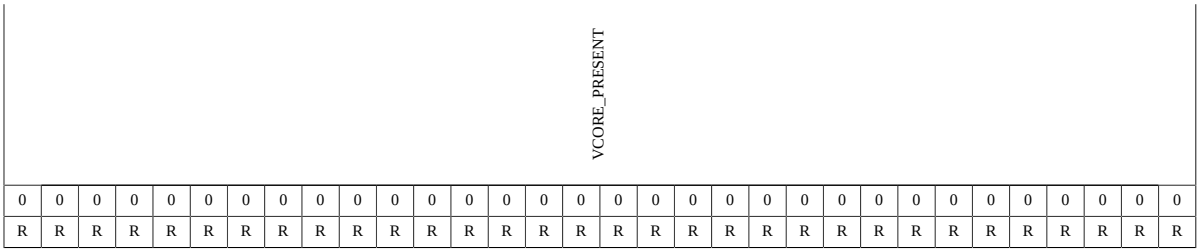


Table 1.107. VPU_RET_VCORE_PRESET Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	VCORE_PRESENT	R	Number of VCOREs present (turn on cores)	0x0

1.2.4.1.45. ENC_PIC_SUB_FRAME_SYNC_IF (0x00000300)

Table 1.108. ENC_PIC_SUB_FRAME_SYNC_IF Bit Assignment

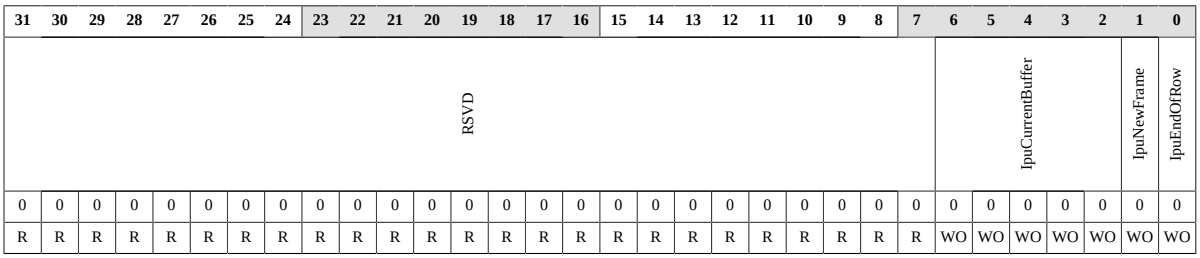


Table 1.109. ENC_PIC_SUB_FRAME_SYNC_IF Field Description

Bit	Name	Type	Function	Reset Value
[31:7]	RSVD	R	Reserved	0x0
[6:2]	IpuCurrentBuffer	WO	An index of source frame buffer where source frame is being uploaded (max: 5) 5'b00001 : Source frame buffer 0 5'b00010 : Source frame buffer 1 5'b00100 : Source frame buffer 2 5'b01000 : Source frame buffer 3 5'b10000 : Source frame buffer 4	0x0
[1]	IpuNewFrame	WO	This bit is toggled whenever IPU starts loading a new frame like 0(default), 1, 0, 1 and so forth.	0x0
[0]	IpuEndOfRow	WO	This bit is toggled whenever IPU finishes writing 64 pixel rows like 0(default), 1, 0, 1 and so forth . This register is valid when register-based Subframe Sync is enabled through CMD_CREATE_INST_SUB_FRM_SYNC[1]. **Note** For more information about the register based Subframe Sync, please refer to the datasheet.	0x0

1.2.4.2. Command I/O Register Descriptions

1.2.4.2.1. Common Parameter Registers

These are the I/O parameter registers which are defined in common for all the commands. Host processor can set arguments or check return values on these registers regardless of the command type.

1.2.4.2.1.1. COMMAND (0x00000100)

Command to run VPU
Depending on the value of command, host interface registers in 0x104 and from 0x114 to 0x1DC might mean different things.

Table 1.110. COMMAND Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMMAND																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.111. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	RW	0x0001 : INIT_VPU 0x0002 : WAKEUP_VPU 0x0004 : SLEEP_VPU 0x0008 : CREATE_INST 0x0010 : FLUSH_INST 0x0020 : DESTROY_INST 0x0040 : INIT_SEQ 0x0080 : SET_FB 0x0100 : ENC_PIC / DEC_PIC 0x0200: ENC_SET_PARAM 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.4.2.1.2. CMD_OPTION (0x00000104)

Option for the command. Details of the option will be described in register sheet for each command.
If the register is not presented in the command sheet, CMD_OPTION is not used for the command.

Table 1.112. CMD_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.113. CMD_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R/W	Reserved	0x0

1.2.4.2.1.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.114. RET_SUCCESS Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																														RUN_CMD_STATUS	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W

Table 1.115. RET_SUCCESS Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1:0]	RUN_CMD_STATUS	R/W	00: FAIL 01: SUCCESS 10: SUCCESS_WITH_WARNING	0x0

1.2.4.2.1.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.116. RET_FAIL_REASON Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_REASON																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.117. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section A.1, "System Error"* defining errors that VPU might have.	0x0

1.2.4.2.1.5. CMD_INSTANCE_INFO (0x00000110)

Instance information

Table 1.118. CMD_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODEC_STD																INST_INDEX															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.119. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run STD_HEVC_DEC = 0x00 STD_HEVC_ENC = 0x01 STD_AVC_DEC = 0x02 STD_AVC_ENC = 0x03 STD_VP9_DEC = 0x16 STD_AVS2_DEC = 0x18 STD_SVAC_DEC = 0x20 STD_SVAC_ENC = 0x21	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can encode or decode more than one instance simultaneously. If multiple instances are running, each instance must have its own process index that is assigned by this register. For example, two instances are running simultaneously, the first instance has the process index 0, the second instance has the process index 1. Instance index shall be in the range of 0 to 31, inclusive.	0x0

1.2.4.2.1.6. RET_QUEUE_STATUS (0x000001E0)

Table 1.120. RET_QUEUE_STATUS Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								QUEUED_COMMAND_NUM_CURRENT_ISNT								QUEUED_COMMAND_NUM															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.121. RET_QUEUE_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	RSVD	R	Reserved	0x0
[23:16]	QUEUED_COMMAND_NUM_CURRENT_ISNT	R/W	The number of queued command for the current instance	0x0
[15:0]	QUEUED_COMMAND_NUM	R/W	The number of queued command	0x0

1.2.4.2.1.7. RET_BS_EMPTY (0x000001E4)

Valid when CPB empty interrupt is asserted(Decoder Only)

Table 1.122. RET_BS_EMPTY Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS_EMPTY_INST_FLAG																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.123. RET_BS_EMPTY Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_EMPTY_INST_FLAG	R/W	BS_EMPTY_INST_FLAG = 1 << instance_index When bitstream data in CPB is not sufficient to completes INIT_SEQ command or DEC_PIC command of an instance, VPU triggers an interrupt to host to request feeding additional bistream data with the instance index. Host can identify which instance's CPB is in empty status with instance_index. This field is ignored after BS_EMPTY interrupt service is done. NOTE: This field is not asserted for the encoder instances.	0x0

1.2.4.2.1.8. RET_QUEUED_CMD_DONE (0x000001E8)

Valid when Queued command done

Table 1.124. RET_QUEUED_CMD_DONE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QUEUED_CMD_INST_FLAG																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.125. RET_QUEUED_CMD_DONE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	QUEUED_CMD_INST_FLAG	R/W	QUEUED_CMD_INST_FLAG = 1 << instance_index. When VPU completes internal processing of queueable command, such as ENC_SET_PARAM, ENC_PIC command for encoder instance and INIT_SEQ, DEC_PIC command for decoder instance, is ready to output the processing result to host, VPU triggers an interrupt to host with the in-	0x0

Bit	Name	Type	Function	Reset Value
			stance index. Host can recieve the output result information of the instance indicated by intance_index. This field is ignored after handling interrupt service for the queueable command.	

1.2.4.2.1.9. RET_SEEK_INSTANCE_INFO (0x000001EC)

A working instance index on seek stage (for internal use only)

Table 1.126. RET_SEEK_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								SEEK_INST_ID							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.127. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INST_ID	R	An instance index on the seek stage of command queue	0x0

1.2.4.2.1.10. RET_PARSING_INSTANCE_INFO (0x000001F0)

A working instance index on prescan stage (for internal use only)

Table 1.128. RET_PARSING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRES_INST_ID_CORE3								PRES_INST_ID_CORE2								PRES_INST_ID_CORE1								PRES_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.129. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PRES_INST_ID_CORE3	R	An instance index on prescan core #3	0x0
[23:16]	PRES_INST_ID_CORE2	R	An instance index on prescan core #2	0x0
[15:8]	PRES_INST_ID_CORE1	R	An instance index on prescan core #1	0x0
[7:0]	PRES_INST_ID_CORE0	R	An instance index on prescan core #0	0x0

1.2.4.2.1.11. RET_DECODING_INSTANCE_INFO (0x000001F4)

A working instance index on decoding stage (for internal use only)

Table 1.130. RET_DECODING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEC_INST_ID_CORE3								DEC_INST_ID_CORE2								DEC_INST_ID_CORE1								DEC_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.131. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	DEC_INST_ID_CORE3	R	An instance index on vcore core #3	0x0
[23:16]	DEC_INST_ID_CORE2	R	An instance index on vcore core #2	0x0
[15:8]	DEC_INST_ID_CORE1	R	An instance index on vcore core #1	0x0
[7:0]	DEC_INST_ID_CORE0	R	An instance index on vcore core #0	0x0

1.2.4.2.1.12. RET_ENCODING_INSTANCE_INFO (0x000001F8)

A working instance index on packing stage (for internal use only)

Table 1.132. RET_ENCODING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PACK_INST_ID_CORE3								PACK_INST_ID_CORE2								PACK_INST_ID_CORE1								PACK_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.133. RET_ENCODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PACK_INST_ID_CORE3	R	An instance index on packing core #3 (reserved for encoder)	0x0
[23:16]	PACK_INST_ID_CORE2	R	An instance index on packing core #2 (reserved for encoder)	0x0
[15:8]	PACK_INST_ID_CORE1	R	An instance index on packing core #1 (reserved for encoder)	0x0
[7:0]	PACK_INST_ID_CORE0	R	An instance index on packing core #0 (reserved for encoder)	0x0

1.2.4.2.1.13. RET_DONE_INSTANCE_INFO (0x000001FC)

Picture done interrupt instance index

Table 1.134. RET_DONE_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											DONE_INST_IDX					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W

Table 1.135. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	DONE_INST_IDX	R/W	Picture done interrupt instance index	0x0

1.2.4.2.2. INIT_VPU Command Parameter Registers

These are command I/O registers where Host processor can set arguments for the INIT_VPU command or get return values.

1.2.4.2.2.1. ADDR_CODE_BASE (0x00000110)

Code buffer base address

Table 1.136. ADDR_CODE_BASE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODE_BUF_BASE																				RSVD											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.137. ADDR_CODE_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:12]	CODE_BUF_BASE	R/W	Base address of V-CPU micro code buffer It should be aligned to 4KB range.	0x0
[11:0]	RSVD	R	Reserved	0x0

1.2.4.2.2.2. CODE_SIZE (0x00000114)

Code buffer size

Table 1.138. CODE_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODE_BUF_SIZE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.139. CODE_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CODE_BUF_SIZE	R/W	Size of CODE buffer It should be aligned to 4KB range.	0x0

1.2.4.2.2.3. CODE_PARAM (0x00000118)

Code buffer parameters

Table 1.140. CODE_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD																								CODE_AXIID				CODE_ENDIAN			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.141. CODE_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:4]	CODE_AXIID	R/W	AXI-ID for the V-CPU part (for virtualization)	0x0
[3:0]	CODE_ENDIAN	R/W	Endianness	0x0

1.2.4.2.2.4. CMD_INIT_ADDR_TEMP_BASE (0x0000011C)

Temporal buffer base address

Table 1.142. CMD_INIT_ADDR_TEMP_BASE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEMP_BUF_BASE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.143. CMD_INIT_ADDR_TEMP_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	TEMP_BUF_BASE	R/W	Base address of temporal buffer for this frame **Note** Each V-CORE should set this field for its own temporal buffer.	0x0

1.2.4.2.2.5. CMD_INIT_TEMP_SIZE (0x00000120)

Temporal buffer size

Table 1.144. CMD_INIT_TEMP_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEMP_BUF_SIZE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.145. CMD_INIT_TEMP_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	TEMP_BUF_SIZE	R/W	Temporal buffer size for the frame	0x0

1.2.4.2.2.6. CMD_INIT_ADDR_SEC_AXI (0x00000124)

Secondary AXI base address
It is valid only when USE_SEC_AXI is not 0.

Table 1.146. CMD_INIT_ADDR_SEC_AXI Bit Assignment

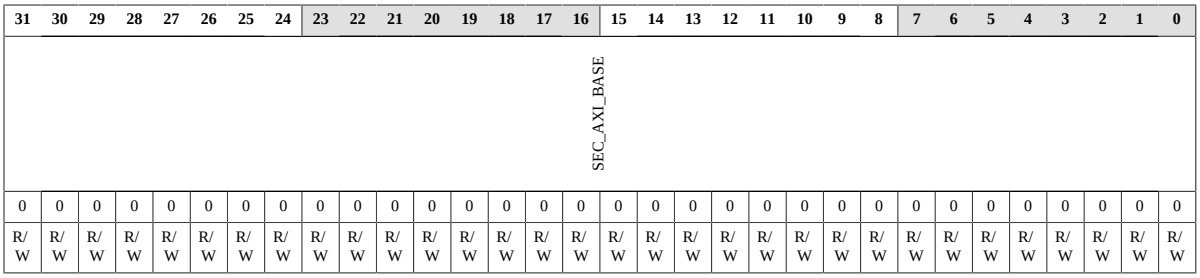


Table 1.147. CMD_INIT_ADDR_SEC_AXI Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SEC_AXI_BASE	R/W	The base address of secondary AXI memory	0x0

1.2.4.2.2.7. CMD_INIT_SEC_AXI_SIZE (0x00000128)

Seconary AXI memory size
It is valid only when USE_SEC_AXI is not 0.

Table 1.148. CMD_INIT_SEC_AXI_SIZE Bit Assignment

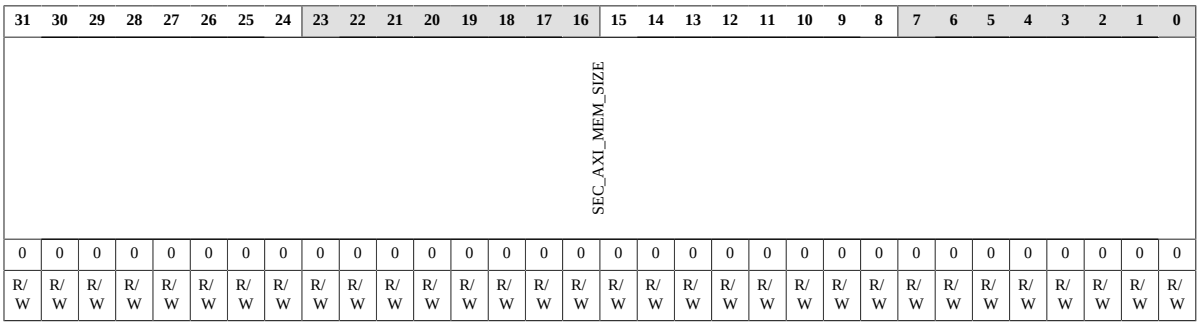


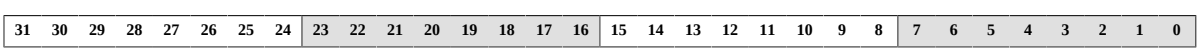
Table 1.149. CMD_INIT_SEC_AXI_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SEC_AXI_MEM_SIZE	R/W	The size of secondary AXI temporal buffer	0x0

1.2.4.2.2.8. CMD_INIT_HW_OPTION (0x0000012C)

VPU hardware option

Table 1.150. CMD_INIT_HW_OPTION Bit Assignment



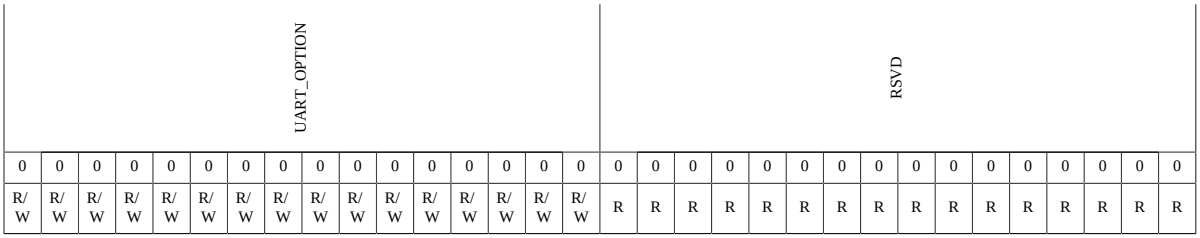


Table 1.151. CMD_INIT_HW_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	UART_OPTION	R/W	UART Divisor	0x0
[15:0]	RSVD	R	Reserved	0x0

1.2.4.2.2.9. CMD_WAKEUP_SYSTEM_CLOCK (0x00000130)

Time out count

Table 1.152. CMD_WAKEUP_SYSTEM_CLOCK Bit Assignment

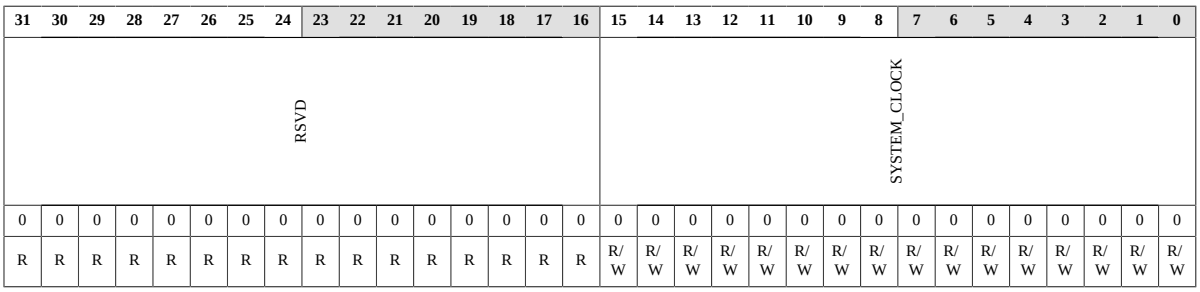


Table 1.153. CMD_WAKEUP_SYSTEM_CLOCK Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:0]	SYSTEM_CLOCK	R/W	System clock information for checking watchdog timer (Reserved)	0x0

1.2.4.2.2.10. CMD_INIT_NUM_TASK_BUF (0x00000134)

Number of task buffer

Table 1.154. CMD_INIT_NUM_TASK_BUF Bit Assignment

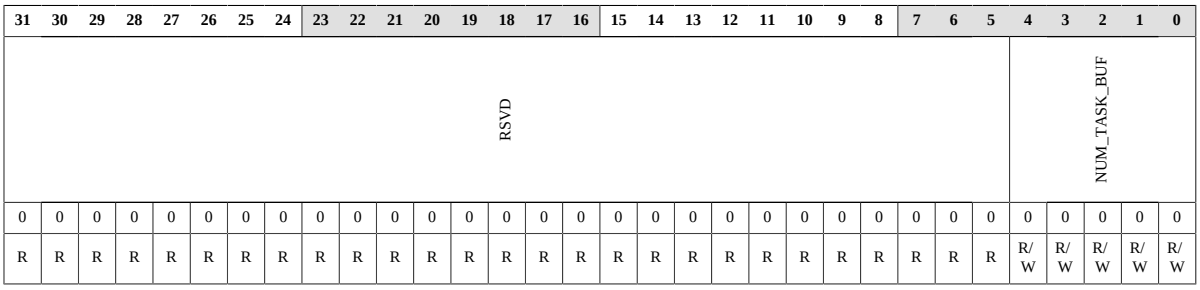


Table 1.155. CMD_INIT_NUM_TASK_BUF Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[4:0]	NUM_TASK_BUF	R/W	Number of valid task buffer (for max queueing depth)	0x0

1.2.4.2.2.11. CMD_INIT_ADDR_TASK_BUF0 (0x00000138)

Table 1.156. CMD_INIT_ADDR_TASK_BUF0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.157. CMD_INIT_ADDR_TASK_BUF0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF0	R/W	Base address of the 1st task buffer	0x0

1.2.4.2.2.12. CMD_INIT_ADDR_TASK_BUF1 (0x0000013C)

Table 1.158. CMD_INIT_ADDR_TASK_BUF1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF1																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.159. CMD_INIT_ADDR_TASK_BUF1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF1	R/W	Base address of the 2ndt task buffer	0x0

1.2.4.2.2.13. CMD_INIT_ADDR_TASK_BUF2 (0x00000140)

Table 1.160. CMD_INIT_ADDR_TASK_BUF2 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF2																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.161. CMD_INIT_ADDR_TASK_BUF2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF2	R/W	Base address of the 3rd task buffer	0x0

1.2.4.2.2.14. CMD_INIT_ADDR_TASK_BUF3 (0x00000144)

Table 1.162. CMD_INIT_ADDR_TASK_BUF3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF3																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.163. CMD_INIT_ADDR_TASK_BUF3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF3	R/W	Base address of the 4th task buffer	0x0

1.2.4.2.2.15. CMD_INIT_ADDR_TASK_BUF4 (0x00000148)

Table 1.164. CMD_INIT_ADDR_TASK_BUF4 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF4																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.165. CMD_INIT_ADDR_TASK_BUF4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF4	R/W	Base address of the 5th task buffer	0x0

1.2.4.2.2.16. CMD_INIT_ADDR_TASK_BUF5 (0x0000014C)

Table 1.166. CMD_INIT_ADDR_TASK_BUF5 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF5																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.167. CMD_INIT_ADDR_TASK_BUF5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF5	R/W	Base address of the 6th task buffer	0x0

1.2.4.2.2.17. CMD_INIT_ADDR_TASK_BUF6 (0x00000150)

Table 1.168. CMD_INIT_ADDR_TASK_BUF6 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF6																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.169. CMD_INIT_ADDR_TASK_BUF6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF6	R/W	Base address of the 7th task buffer	0x0

1.2.4.2.2.18. CMD_INIT_ADDR_TASK_BUF7 (0x00000154)

Table 1.170. CMD_INIT_ADDR_TASK_BUF7 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF7																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.171. CMD_INIT_ADDR_TASK_BUF7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF7	R/W	Base address of the 8th task buffer	0x0

1.2.4.2.2.19. CMD_INIT_ADDR_TASK_BUF8 (0x00000158)

Table 1.172. CMD_INIT_ADDR_TASK_BUF8 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF8																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.173. CMD_INIT_ADDR_TASK_BUF8 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF8	R/W	Base address of the 9th task buffer	0x0

1.2.4.2.2.20. CMD_INIT_ADDR_TASK_BUF9 (0x0000015C)

Table 1.174. CMD_INIT_ADDR_TASK_BUF9 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF9																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.175. CMD_INIT_ADDR_TASK_BUF9 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF9	R/W	Base address of the 10th task buffer	0x0

1.2.4.2.2.21. CMD_INIT_ADDR_TASK_BUFA (0x00000160)

Table 1.176. CMD_INIT_ADDR_TASK_BUFA Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFA																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.177. CMD_INIT_ADDR_TASK_BUFA Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFA	R/W	Base address of the 11th task buffer	0x0

1.2.4.2.2.22. CMD_INIT_ADDR_TASK_BUFB (0x00000164)

Table 1.178. CMD_INIT_ADDR_TASK_BUFB Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFB																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.179. CMD_INIT_ADDR_TASK_BUFB Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFB	R/W	Base address of the 12th task buffer	0x0

1.2.4.2.2.23. CMD_INIT_ADDR_TASK_BUFC (0x00000168)

Table 1.180. CMD_INIT_ADDR_TASK_BUFC Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFC																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.181. CMD_INIT_ADDR_TASK_BUFC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFC	R/W	Base address of the 13th task buffer	0x0

1.2.4.2.2.24. CMD_INIT_ADDR_TASK_BUFD (0x0000016C)

Table 1.182. CMD_INIT_ADDR_TASK_BUFD Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.183. CMD_INIT_ADDR_TASK_BUFD Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFD	R/W	Base address of the 14th task buffer	0x0

1.2.4.2.2.25. CMD_INIT_ADDR_TASK_BUFE (0x00000170)

Table 1.184. CMD_INIT_ADDR_TASK_BUFE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFE																															

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.185. CMD_INIT_ADDR_TASK_BUFE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFE	R/W	Base address of the 15th task buffer	0x0

1.2.4.2.2.26. CMD_INIT_ADDR_TASK_BUFF (0x00000174)

Table 1.186. CMD_INIT_ADDR_TASK_BUFF Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFF																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.187. CMD_INIT_ADDR_TASK_BUFF Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFF	R/W	Base address of the 16th task buffer	0x0

1.2.4.2.2.27. CMD_INIT_TASK_BUFF_SIZE (0x00000178)

Table 1.188. CMD_INIT_TASK_BUFF_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TASK_BUFF_SIZE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.189. CMD_INIT_TASK_BUFF_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	TASK_BUFF_SIZE	R/W	The size of task buffer	0x0

1.2.4.2.3. WAKEUP_VPU Command Parameter Registers

These are command I/O registers where Host processor can set arguments for the WAKEUP_VPU command or get return values.

1.2.4.2.3.1. CMD_WAKEUP_ADDR_CODE_BASE (0x00000110)

Code buffer base address

Table 1.190. CMD_WAKEUP_ADDR_CODE_BASE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODE_BUF_BASE																				RSVD											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.191. CMD_WAKEUP_ADDR_CODE_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:12]	CODE_BUF_BASE	R/W	Base address of V-CPU micro code buffer It should be aligned to 4KB range.	0x0
[11:0]	RSVD	R	Reserved	0x0

1.2.4.2.3.2. CMD_WAKEUP_CODE_SIZE (0x00000114)

Code buffer size

Table 1.192. CMD_WAKEUP_CODE_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODE_BUF_SIZE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.193. CMD_WAKEUP_CODE_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CODE_BUF_SIZE	R/W	Size of CODE buffer It should be aligned to 4KB range.	0x0

1.2.4.2.3.3. CMD_WAKEUP_CODE_PARAM (0x00000118)

Code buffer parameter

Table 1.194. CMD_WAKEUP_CODE_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

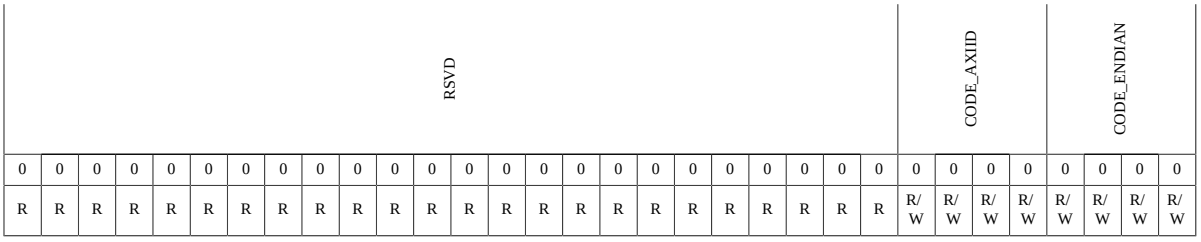


Table 1.195. CMD_WAKEUP_CODE_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:4]	CODE_AXIID	R/W	AXI-ID for the V-CPU part (for virtualization)	0x0
[3:0]	CODE_ENDIAN	R/W	Endianness	0x0

1.2.4.2.3.4. CMD_WAKEUP_ADDR_TEMP_BASE (0x0000011C)

Temporal buffer base address

Table 1.196. CMD_WAKEUP_ADDR_TEMP_BASE Bit Assignment

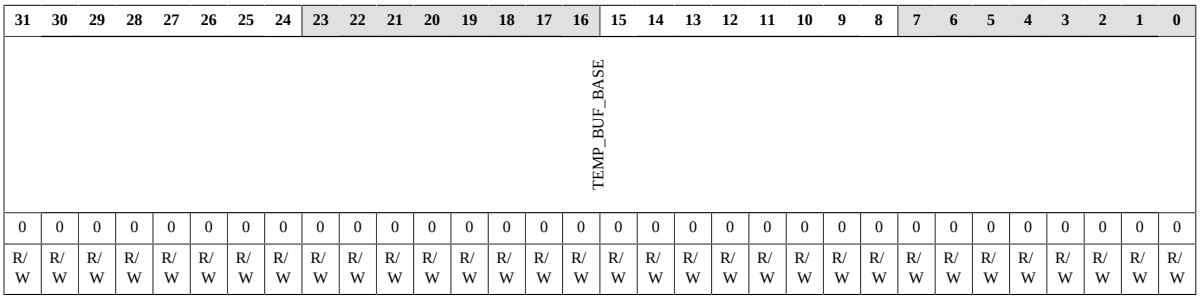


Table 1.197. CMD_WAKEUP_ADDR_TEMP_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	TEMP_BUF_BASE	R/W	Base address of temporal buffer for this frame **Note** Each V-CORE should set this field for its own temporal buffer.	0x0

1.2.4.2.3.5. CMD_WAKEUP_TEMP_SIZE (0x00000120)

Temporal buffer size

Table 1.198. CMD_WAKEUP_TEMP_SIZE Bit Assignment

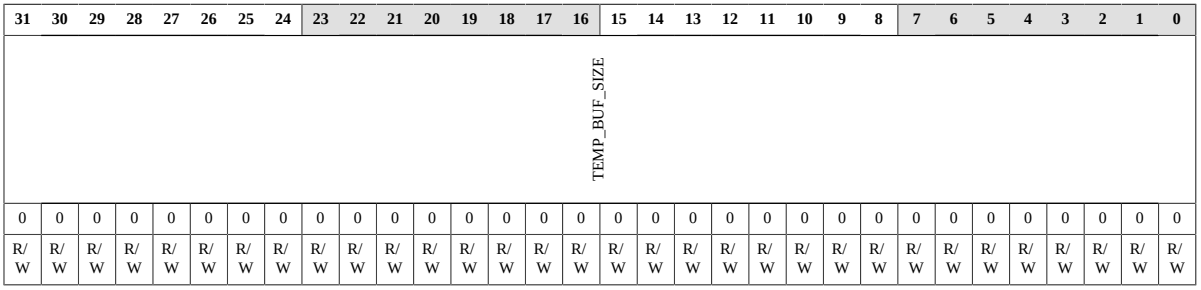


Table 1.199. CMD_WAKEUP_TEMP_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	TEMP_BUF_SIZE	R/W	Temporal buffer size for this frame	0x0

1.2.4.2.3.6. CMD_WAKEUP_ADDR_SEC_AXI (0x00000124)

Secondary AXI base address

Table 1.200. CMD_WAKEUP_ADDR_SEC_AXI Bit Assignment

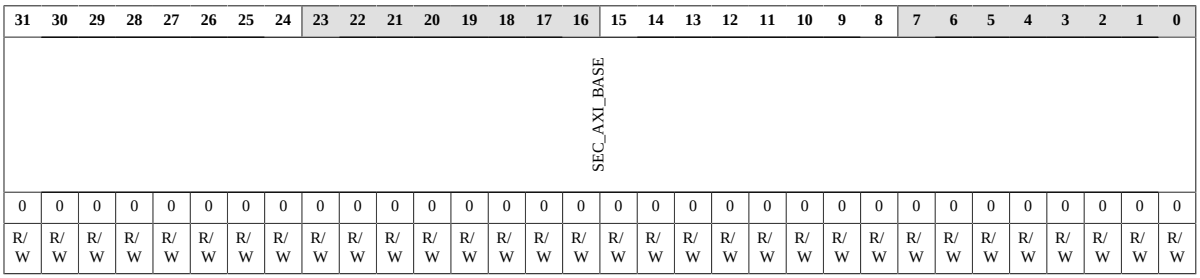


Table 1.201. CMD_WAKEUP_ADDR_SEC_AXI Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SEC_AXI_BASE	R/W	The base address of secondary AXI memory It is valid only when USE_SEC_AXI is not 0.	0x0

1.2.4.2.3.7. CMD_WAKEUP_SEC_AXI_SIZE (0x00000128)

Seconary AXI memory size

Table 1.202. CMD_WAKEUP_SEC_AXI_SIZE Bit Assignment

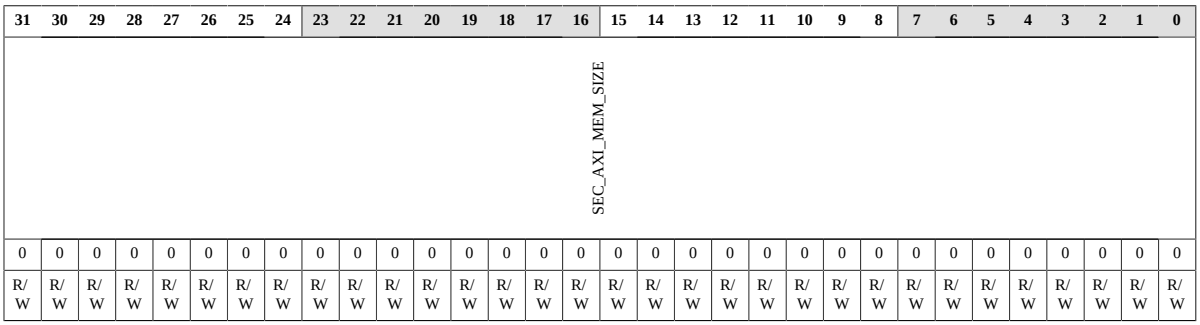


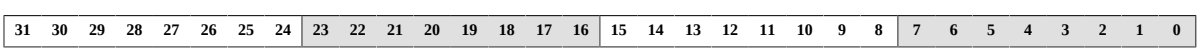
Table 1.203. CMD_WAKEUP_SEC_AXI_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SEC_AXI_MEM_SIZE	R/W	The size of secondary AXI temporal buffer It is valid only when USE_SEC_AXI is not 0.	0x0

1.2.4.2.3.8. CMD_WAKEUP_HW_OPTION (0x0000012C)

VPU hardware option

Table 1.204. CMD_WAKEUP_HW_OPTION Bit Assignment



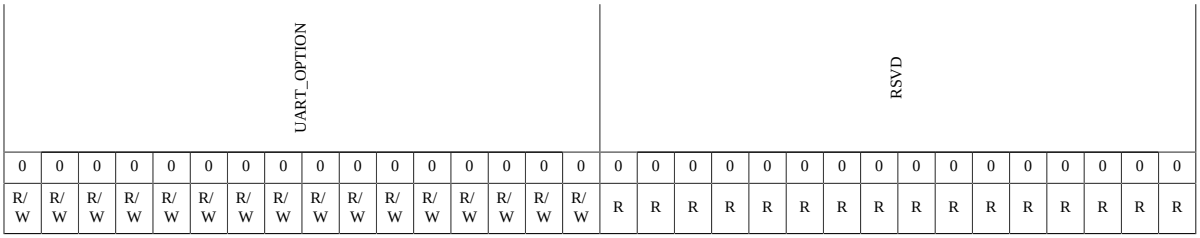


Table 1.205. CMD_WAKEUP_HW_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	UART_OPTION	R/W	UART Divisor	0x0
[15:0]	RSVD	R	Reserved	0x0

1.2.4.2.3.9. CMD_WAKEUP_SYSTEM_CLOCK (0x00000130)

Time out count

Table 1.206. CMD_WAKEUP_SYSTEM_CLOCK Bit Assignment

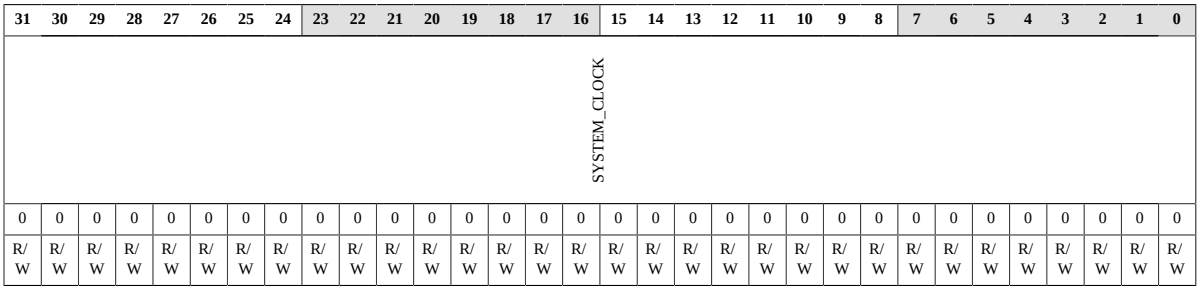


Table 1.207. CMD_WAKEUP_SYSTEM_CLOCK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SYSTEM_CLOCK	R/W	System clock information for checking Watchdog timer	0x0

1.2.4.2.3.10. CMD_WAKEUP_NUM_TASK_BUF (0x00000134)

Number of task buffer

Table 1.208. CMD_WAKEUP_NUM_TASK_BUF Bit Assignment

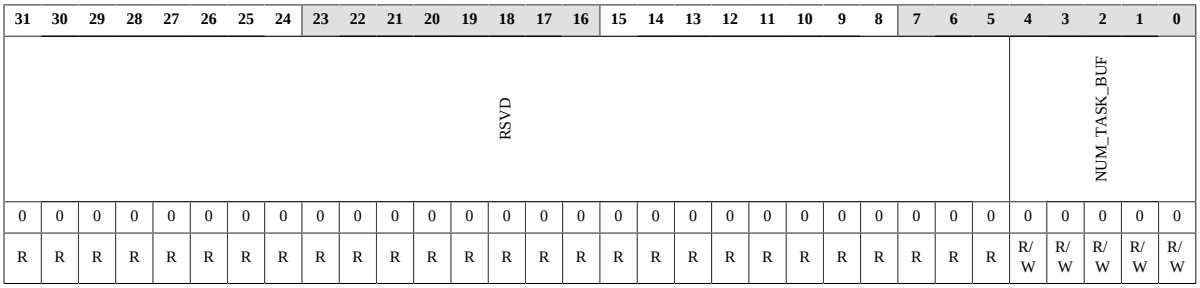


Table 1.209. CMD_WAKEUP_NUM_TASK_BUF Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[4:0]	NUM_TASK_BUF	R/W	Number of valid task buffer (for max queueing depth)	0x0

1.2.4.2.3.11. CMD_WAKEUP_ADDR_TASK_BUF0 (0x00000138)

Table 1.210. CMD_WAKEUP_ADDR_TASK_BUF0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.211. CMD_WAKEUP_ADDR_TASK_BUF0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF0	R/W	Base address of the 1st task buffer	0x0

1.2.4.2.3.12. CMD_WAKEUP_ADDR_TASK_BUF1 (0x0000013C)

Table 1.212. CMD_WAKEUP_ADDR_TASK_BUF1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF1																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.213. CMD_WAKEUP_ADDR_TASK_BUF1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF1	R/W	Base address of the 2ndt task buffer	0x0

1.2.4.2.3.13. CMD_WAKEUP_ADDR_TASK_BUF2 (0x00000140)

Table 1.214. CMD_WAKEUP_ADDR_TASK_BUF2 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF2																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.215. CMD_WAKEUP_ADDR_TASK_BUF2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF2	R/W	Base address of the 3rd task buffer	0x0

1.2.4.2.3.14. CMD_WAKEUP_ADDR_TASK_BUF3 (0x00000144)

Table 1.216. CMD_WAKEUP_ADDR_TASK_BUF3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF3																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.217. CMD_WAKEUP_ADDR_TASK_BUF3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF3	R/W	Base address of the 4th task buffer	0x0

1.2.4.2.3.15. CMD_WAKEUP_ADDR_TASK_BUF4 (0x00000148)

Table 1.218. CMD_WAKEUP_ADDR_TASK_BUF4 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF4																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.219. CMD_WAKEUP_ADDR_TASK_BUF4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF4	R/W	Base address of the 5th task buffer	0x0

1.2.4.2.3.16. CMD_WAKEUP_ADDR_TASK_BUF5 (0x0000014C)

Table 1.220. CMD_WAKEUP_ADDR_TASK_BUF5 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF5																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.221. CMD_WAKEUP_ADDR_TASK_BUF5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF5	R/W	Base address of the 6th task buffer	0x0

1.2.4.2.3.17. CMD_WAKEUP_ADDR_TASK_BUF6 (0x00000150)

Table 1.222. CMD_WAKEUP_ADDR_TASK_BUF6 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF6																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.223. CMD_WAKEUP_ADDR_TASK_BUF6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF6	R/W	Base address of the 7th task buffer	0x0

1.2.4.2.3.18. CMD_WAKEUP_ADDR_TASK_BUF7 (0x00000154)

Table 1.224. CMD_WAKEUP_ADDR_TASK_BUF7 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF7																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.225. CMD_WAKEUP_ADDR_TASK_BUF7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF7	R/W	Base address of the 8th task buffer	0x0

1.2.4.2.3.19. CMD_WAKEUP_ADDR_TASK_BUF8 (0x00000158)

Table 1.226. CMD_WAKEUP_ADDR_TASK_BUF8 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF8																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.227. CMD_WAKEUP_ADDR_TASK_BUF8 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF8	R/W	Base address of the 9th task buffer	0x0

1.2.4.2.3.20. CMD_WAKEUP_ADDR_TASK_BUF9 (0x0000015C)

Table 1.228. CMD_WAKEUP_ADDR_TASK_BUF9 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUF9																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.229. CMD_WAKEUP_ADDR_TASK_BUF9 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUF9	R/W	Base address of the 10th task buffer	0x0

1.2.4.2.3.21. CMD_WAKEUP_ADDR_TASK_BUFA (0x00000160)

Table 1.230. CMD_WAKEUP_ADDR_TASK_BUFA Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFA																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.231. CMD_WAKEUP_ADDR_TASK_BUFA Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFA	R/W	Base address of the 11th task buffer	0x0

1.2.4.2.3.22. CMD_WAKEUP_ADDR_TASK_BUFB (0x00000164)

Table 1.232. CMD_WAKEUP_ADDR_TASK_BUFB Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFB																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.233. CMD_WAKEUP_ADDR_TASK_BUFB Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFB	R/W	Base address of the 12th task buffer	0x0

1.2.4.2.3.23. CMD_WAKEUP_ADDR_TASK_BUFC (0x00000168)

Table 1.234. CMD_WAKEUP_ADDR_TASK_BUFC Bit Assignment

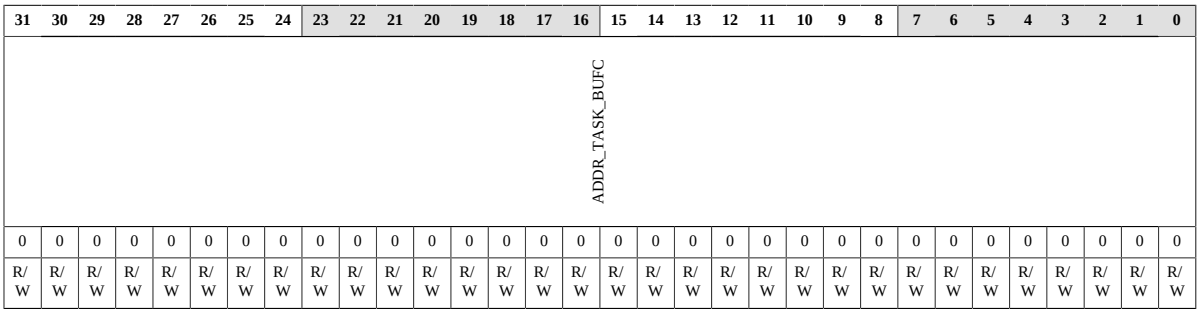


Table 1.235. CMD_WAKEUP_ADDR_TASK_BUFC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFC	R/W	Base address of the 13th task buffer	0x0

1.2.4.2.3.24. CMD_WAKEUP_ADDR_TASK_BUFD (0x0000016C)

Table 1.236. CMD_WAKEUP_ADDR_TASK_BUFD Bit Assignment

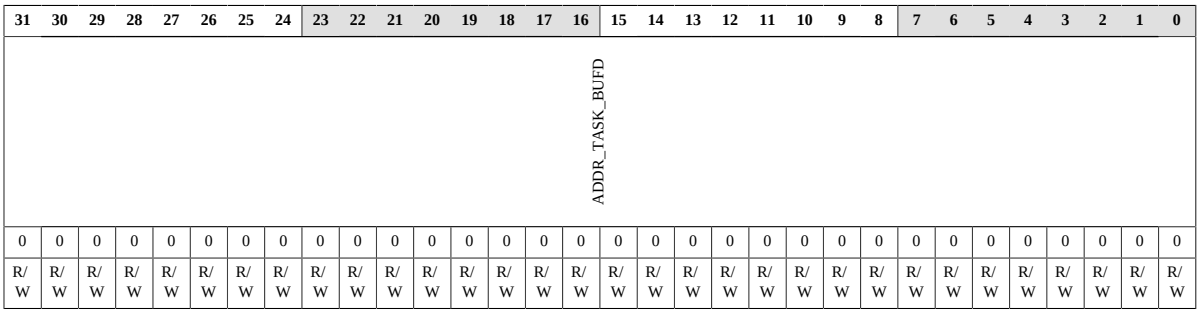
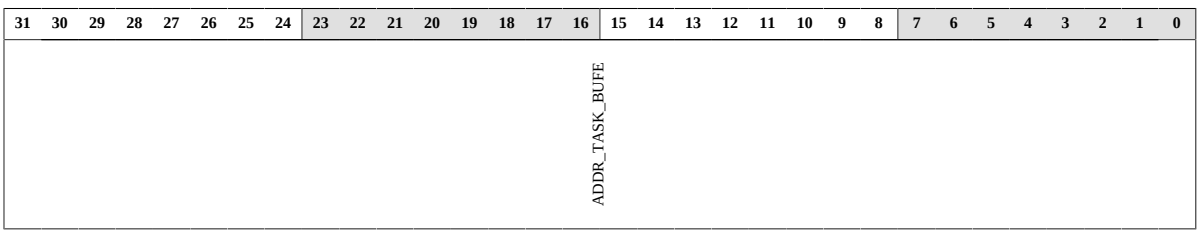


Table 1.237. CMD_WAKEUP_ADDR_TASK_BUFD Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFD	R/W	Base address of the 14th task buffer	0x0

1.2.4.2.3.25. CMD_WAKEUP_ADDR_TASK_BUFE (0x00000170)

Table 1.238. CMD_WAKEUP_ADDR_TASK_BUFE Bit Assignment



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.239. CMD_WAKEUP_ADDR_TASK_BUFE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFE	R/W	Base address of the 15th task buffer	0x0

1.2.4.2.3.26. CMD_WAKEUP_ADDR_TASK_BUFF (0x00000174)

Table 1.240. CMD_WAKEUP_ADDR_TASK_BUFF Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TASK_BUFF																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.241. CMD_WAKEUP_ADDR_TASK_BUFF Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_TASK_BUFF	R/W	Base address of the 16th task buffer	0x0

1.2.4.2.3.27. CMD_WAKEUP_TASK_BUFF_SIZE (0x00000178)

Table 1.242. CMD_WAKEUP_TASK_BUFF_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TASK_BUFF_SIZE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.243. CMD_WAKEUP_TASK_BUFF_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	TASK_BUFF_SIZE	R/W	The size of task buffer	0x0

1.2.4.2.4. CREATE_INST Command Parameter Registers for Encoder

These are command I/O registers where Host processor can set arguments for the CREATE_INST command or get return values.

1.2.4.2.4.1. CMD_CREATE_INST_ADDR_WORK_BASE (0x00000114)

Work buffer base address

Table 1.244. CMD_CREATE_INST_ADDR_WORK_BASE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WORK_BUF_BASE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.245. CMD_CREATE_INST_ADDR_WORK_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	WORK_BUF_BASE	R/W	Base address of work buffer for each instance This address should be aligned in 4KB boundary.	0x0

1.2.4.2.4.2. CMD_CREATE_INST_WORK_SIZE (0x00000118)

Work buffer size

Table 1.246. CMD_CREATE_INST_WORK_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WORK_BUF_SIZE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.247. CMD_CREATE_INST_WORK_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	WORK_BUF_SIZE	R/W	Size of work buffer (4KB boundary) The size of work buffer should be larger than required minimum work buffer size. This address should be aligned in 4KB boundary.	0x0

1.2.4.2.4.3. CMD_CREATE_INST_SUB_FRM_SYNC (0x00000128)

Enable subframe synchronization

Table 1.248. CMD_CREATE_INST_SUB_FRM_SYNC Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

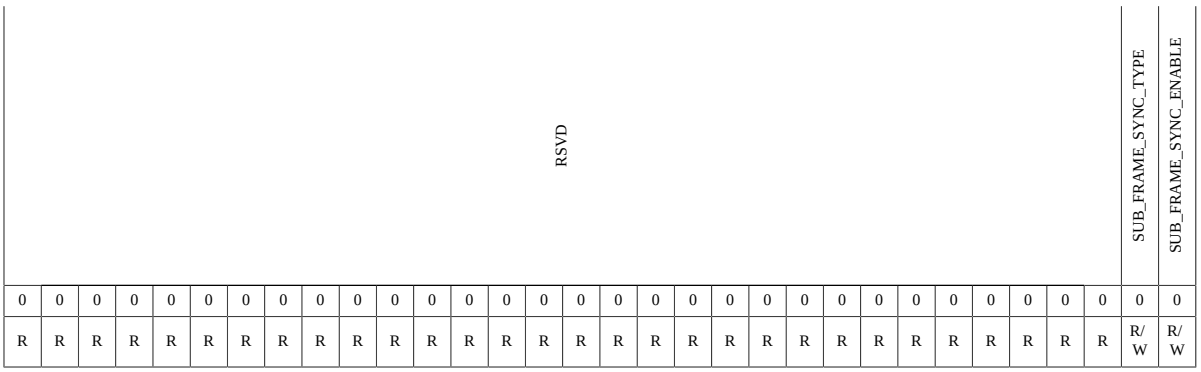


Table 1.249. CMD_CREATE_INST_SUB_FRM_SYNC Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1]	SUB_FRAME_SYNC_TYPE	R/W	0 : hard wired control 1 : register based control	0x0
[0]	SUB_FRAME_SYNC_ENABLE	R/W	0 : don't use subframe synchronization for the instance. 1 : use subframe subframe synchronization for the instance.	0x0

1.2.4.2.5. FLUSH_INST Command Parameter Registers

These are command I/O registers where Host processor can set arguments for the FLUSH_INST command or get return values.

1.2.4.2.5.1. COMMAND (0x00000100)

Command to run VPU
Depending on the value of command, host interface registers from 0x104 to 0x1FC might mean different things. The registers below are associated with **FLUSH_INST** command (**0x0010** is given to this register).

Table 1.250. COMMAND Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMMAND																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.251. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	RW	0x0001 : INIT_VPU 0x0002 : WAKEUP_VPU 0x0004 : SLEEP_VPU 0x0008 : CREATE_INST **0x0010 : FLUSH_INST** 0x0020 : DESTROY_INST 0x0040 : INIT_SEQ 0x0080 : SET_FB 0x0100 : ENC_PIC / DEC_PIC 0x0200 : ENC_SET_PARAM 0x2000 : LOW_LATENCY 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.4.2.5.2. CMD_CREATE_INST_OPTION (0x00000104)

Run command option

Table 1.252. CMD_CREATE_INST_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.253. CMD_CREATE_INST_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R/W	Reserved	0x0

1.2.4.2.5.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.254. RET_SUCCESS Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																														RUN_CMD_STATUS	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W

Table 1.255. RET_SUCCESS Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1:0]	RUN_CMD_STATUS	R/W	00: FAIL 01: SUCCESS 10: SUCCESS_WITH_WARNING	0x0

1.2.4.2.5.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.256. RET_FAIL_REASON Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_REASON																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.257. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section A.1, "System Error"* defining errors that VPU might have.	0x0

1.2.4.2.5.5. CMD_INSTANCE_INFO (0x00000110)

Instance information

Table 1.258. CMD_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODEC_STD																INST_INDEX															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.259. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run STD_HEVC_DEC = 0x00 STD_HEVC_ENC = 0x01 STD_AVC_DEC = 0x02 STD_AVC_ENC = 0x03 STD_VP9_DEC = 0x16 STD_AVS2_DEC = 0x18 STD_SVAC_DEC = 0x20 STD_SVAC_ENC = 0x21	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can encode or decode more than one instance simultaneously. If multiple instances are running, each instance must have its own process index that is assigned by this register. For example, two instances are running simultaneously, the first instance has the process index 0, the second instance has the process index 1. Instance index shall be in the range of 0 to 31, inclusive.	0x0

1.2.4.2.5.6. RET_SEEK_INSTANCE_INFO (0x000001EC)

A working instance index on seek stage (for internal use only)

Table 1.260. RET_SEEK_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								SEEK_INST_ID							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.261. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INST_ID	R	An instance index on the seek stage of command queue	0x0

1.2.4.2.5.7. RET_PARSING_INSTANCE_INFO (0x000001F0)

A working instance index on prescan stage (for internal use only)

Table 1.262. RET_PARSING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRES_INST_ID_CORE3								PRES_INST_ID_CORE2								PRES_INST_ID_CORE1								PRES_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.263. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PRES_INST_ID_CORE3	R	An instance index on prescan core #3	0x0
[23:16]	PRES_INST_ID_CORE2	R	An instance index on prescan core #2	0x0
[15:8]	PRES_INST_ID_CORE1	R	An instance index on prescan core #1	0x0
[7:0]	PRES_INST_ID_CORE0	R	An instance index on prescan core #0	0x0

1.2.4.2.5.8. RET_DECODING_INSTANCE_INFO (0x000001F4)

A working instance index on decoding stage (for internal use only)

Table 1.264. RET_DECODING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEC_INST_ID_CORE3								DEC_INST_ID_CORE2								DEC_INST_ID_CORE1								DEC_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.265. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	DEC_INST_ID_CORE3	R	An instance index on vcore core #3	0x0
[23:16]	DEC_INST_ID_CORE2	R	An instance index on vcore core #2	0x0
[15:8]	DEC_INST_ID_CORE1	R	An instance index on vcore core #1	0x0
[7:0]	DEC_INST_ID_CORE0	R	An instance index on vcore core #0	0x0

1.2.4.2.5.9. RET_ENCODING_INSTANCE_INFO (0x000001F8)

A working instance index on packing stage (for internal use only)

Table 1.266. RET_ENCODING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PACK_INST_ID_CORE3								PACK_INST_ID_CORE2								PACK_INST_ID_CORE1								PACK_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.267. RET_ENCODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PACK_INST_ID_CORE3	R	An instance index on packing core #3 (reserved for encoder)	0x0

Bit	Name	Type	Function	Reset Value
[23:16]	PACK_INST_ID_CORE2	R	An instance index on packing core #2 (reserved for encoder)	0x0
[15:8]	PACK_INST_ID_CORE1	R	An instance index on packing core #1 (reserved for encoder)	0x0
[7:0]	PACK_INST_ID_CORE0	R	An instance index on packing core #0 (reserved for encoder)	0x0

1.2.4.2.6. DESTROY_INST Command Parameter Registers

These are command I/O registers where Host processor can set arguments for the DESTROY_INST command or get return values.

1.2.4.2.6.1. COMMAND (0x00000100)

Command to run VPU
Depending on the value of command, host interface registers from 0x104 to 0x1FC might mean different things. The registers below are associated with **DESTROY_INST** command (**0x0020** is given to this register).

Table 1.268. COMMAND Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMMAND																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.269. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	RW	0x0001 : INIT_VPU 0x0002 : WAKEUP_VPU 0x0004 : SLEEP_VPU 0x0008 : CREATE_INST 0x0010 : FLUSH_INST **0x0020 : DESTROY_INST** 0x0040 : INIT_SEQ 0x0080 : SET_FB 0x0100 : ENC_PIC / DEC_PIC 0x0200 : ENC_SET_PARAM 0x2000 : LOW_LATENCY 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.4.2.6.2. CMD_CREATE_INST_OPTION (0x00000104)

Run command option

Table 1.270. CMD_CREATE_INST_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.271. CMD_CREATE_INST_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R/W	Reserved	0x0

1.2.4.2.6.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.272. RET_SUCCESS Bit Assignment

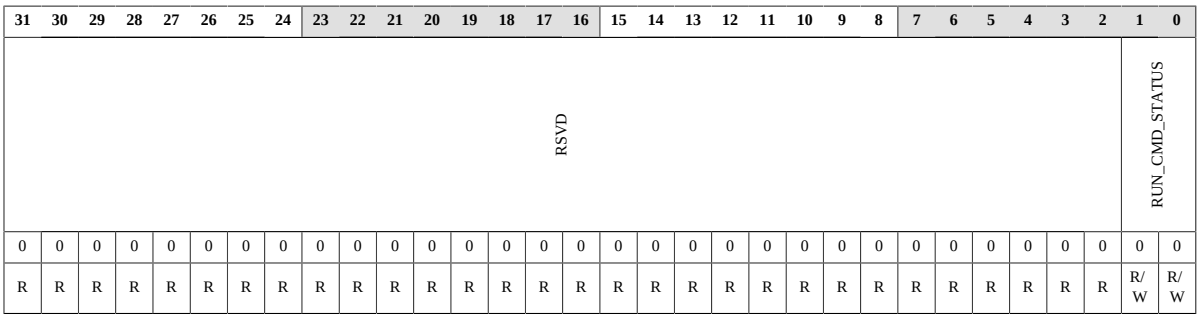


Table 1.273. RET_SUCCESS Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1:0]	RUN_CMD_STATUS	R/W	00: FAIL 01: SUCCESS 10: SUCCESS_WITH_WARNING	0x0

1.2.4.2.6.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.274. RET_FAIL_REASON Bit Assignment

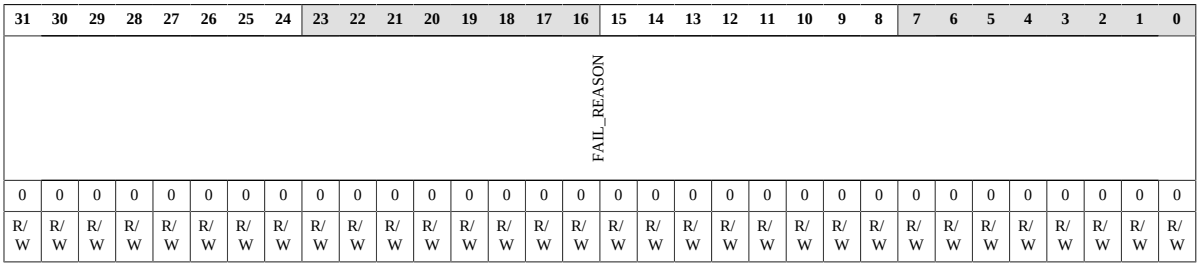


Table 1.275. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section A.1, "System Error"* defining errors that VPU might have.	0x0

1.2.4.2.6.5. CMD_INSTANCE_INFO (0x00000110)

Instance information

Table 1.276. CMD_INSTANCE_INFO Bit Assignment

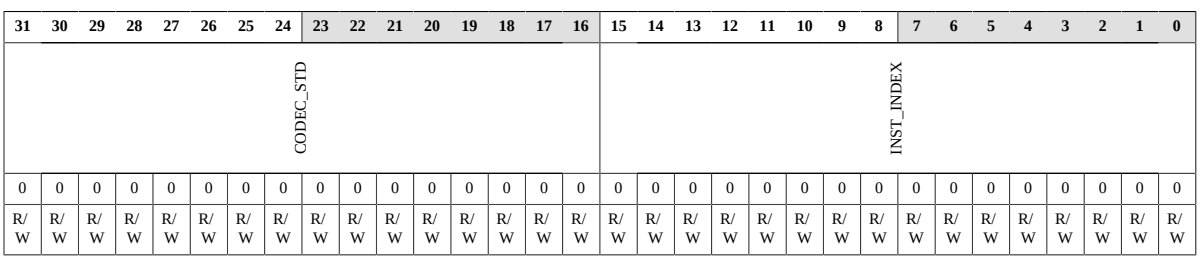


Table 1.277. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run STD_HEVC_DEC = 0x00 STD_HEVC_ENC = 0x01 STD_AVC_DEC = 0x02 STD_AVC_ENC = 0x03 STD_VP9_DEC = 0x16 STD_AVS2_DEC = 0x18 STD_SVAC_DEC = 0x20 STD_SVAC_ENC = 0x21	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can encode or decode more than one instance simultaneously. If multiple instances are running, each instance must have its own process index that is assigned by this register. For example, two instances are running simultaneously, the first instance has the process index 0, the second instance has the process index 1. Instance index shall be in the range of 0 to 31, inclusive.	0x0

1.2.4.2.6.6. RET_SEEK_INSTANCE_INFO (0x000001EC)

A working instance index on seek stage (for internal use only)

Table 1.278. RET_SEEK_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								SEEK_INST_ID							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.279. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INST_ID	R	An instance index on the seek stage of command queue	0x0

1.2.4.2.6.7. RET_PARSING_INSTANCE_INFO (0x000001F0)

A working instance index on prescan stage (for internal use only)

Table 1.280. RET_PARSING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRES_INST_ID_CORE3								PRES_INST_ID_CORE2								PRES_INST_ID_CORE1								PRES_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.281. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PRES_INST_ID_CORE3	R	An instance index on prescan core #3	0x0
[23:16]	PRES_INST_ID_CORE2	R	An instance index on prescan core #2	0x0
[15:8]	PRES_INST_ID_CORE1	R	An instance index on prescan core #1	0x0
[7:0]	PRES_INST_ID_CORE0	R	An instance index on prescan core #0	0x0

1.2.4.2.6.8. RET_DECODING_INSTANCE_INFO (0x000001F4)

A working instance index on decoding stage (for internal use only)

Table 1.282. RET_DECODING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEC_INST_ID_CORE3								DEC_INST_ID_CORE2								DEC_INST_ID_CORE1								DEC_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.283. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	DEC_INST_ID_CORE3	R	An instance index on vcore core #3	0x0
[23:16]	DEC_INST_ID_CORE2	R	An instance index on vcore core #2	0x0
[15:8]	DEC_INST_ID_CORE1	R	An instance index on vcore core #1	0x0
[7:0]	DEC_INST_ID_CORE0	R	An instance index on vcore core #0	0x0

1.2.4.2.6.9. RET_ENCODING_INSTANCE_INFO (0x000001F8)

A working instance index on packing stage (for internal use only)

Table 1.284. RET_ENCODING_INSTANCE_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PACK_INST_ID_CORE3								PACK_INST_ID_CORE2								PACK_INST_ID_CORE1								PACK_INST_ID_CORE0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.285. RET_ENCODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PACK_INST_ID_CORE3	R	An instance index on packing core #3 (reserved for encoder)	0x0

Bit	Name	Type	Function	Reset Value
[23:16]	PACK_INST_ID_CORE2	R	An instance index on packing core #2 (reserved for encoder)	0x0
[15:8]	PACK_INST_ID_CORE1	R	An instance index on packing core #1 (reserved for encoder)	0x0
[7:0]	PACK_INST_ID_CORE0	R	An instance index on packing core #0 (reserved for encoder)	0x0

1.2.4.2.7. ENC_SET_PARAM Command(COMMON) Parameter Registers for H.265/HEVC

There are command I/O registers where Host processor can set arguments for the ENC_SET_PARAM command. These parameters are sent to firmware when SET_PARAM_OPTION(0x104) is set to 0x0, all of which work in the common setting mode.

1.2.4.2.7.1. SET_PARAM_OPTION (0x00000104)

Table 1.286. SET_PARAM_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																														SET_PARAM_OPTION	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W

Table 1.287. SET_PARAM_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1:0]	SET_PARAM_OPTION	R/W	ENC_SET_PARAM command option **0x0 : COMMON** 0x1 : GOP 0x10 : CHANGE_PARAM	0x0

1.2.4.2.7.2. CMD_ENC_SET_PARAM_ENABLE (0x00000118)

Encoder parameter change enable flag

Note These parameter flags are only allowed be changed within the same sequence. In other words, the other encoding parameters such as CMD_ENC_SEQ_SRC_SIZE, CMD_ENC_SEQ_SPS_PARAM, CMD_ENC_SEQ_GOP_PARAM, etc. should never be changed while encoding.

Table 1.288. CMD_ENC_SET_PARAM_ENABLE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									ENABLE_SET_CUSTOM_LAMBDA	ENABLE_SET_CUSTOM_MD	ENABLE_SET_BG_PARAM	ENABLE_SET_NR_PARAM	ENABLE_SET_SEQ_RDO_PARAM	ENABLE_SET_SEQ_DEPENDENT_SLICE	ENABLE_SET_SEQ_INDEPENDENT_SLICE	RSVD					ENABLE_SET_RC_BIT_RATIO_LAYER	ENABLE_SET_RC_MIN_MAX_QP	ENABLE_SET_RC_PARAM	ENABLE_SET_RC_TARGET_RATE	RSVD							ENABLE_SET_SEQ_INTRA_PARAM	ENABLE_SET_PPS_PARAM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.289. CMD_ENC_SET_PARAM_ENABLE Field Description

Bit	Name	Type	Function	Reset Value
[31:23]	RSVD	R	Reserved	0x0
[22]	ENABLE_SET_CUSTOM_LAMBDA	R/W	It enables to change CMD_ENC_CUSTOM_LAMBDA_ADDR.	0x0
[21]	ENABLE_SET_CUSTOM_MD	R/W	It enables to change CMD_ENC_CUSTOM_MD_PU04 to CMD_ENC_CUSTOM_MD_CU32.	0x0
[20]	ENABLE_SET_BG_PARAM	R/W	It enables to change CMD_ENC_BG_PARAM.	0x0
[19]	ENABLE_SET_NR_PARAM	R/W	It enables to change CMD_ENC_NR_PARAM, CMD_ENC_NR_WEIGHT.	0x0
[18]	ENABLE_SET_SEQ_RDO_PARAM	R/W	It enables to change CMD_ENC_SEQ_RDO_PARAM.	0x0
[17]	ENABLE_SET_SEQ_DEPENDENT_SLICE	R/W	It enables to change CMD_ENC_SEQ_DEPENDENT_ SLICE.	0x0
[16]	ENABLE_SET_SEQ_INDEPENDENT_SLICE	R/W	It enables to change CMD_ENC_SEQ_ INDEPENDENT_SLICE.	0x0
[15:12]	RSVD	R/W	Reserved	0x0
[11]	ENABLE_SET_RC_BIT_RATIO_LAYER	R/W	It enables to change CMD_ENC_RC_BIT_RATIO_ LAYER_0_3 and CMD_ENC_RC_BIT_RATIO_ LAYER_4_7.	0x0
[10]	ENABLE_SET_RC_MIN_MAX_QP	R/W	It enables to change CMD_ENC_RC_MIN_MAX_QP and CMD_ENC_RC_INTER_MIN_ MAX_QP.	0x0
[9]	ENABLE_SET_RC_PARAM	R/W	It enables to change CMD_ENC_RC_PARAM.	0x0
[8]	ENABLE_SET_RC_TARGET_RATE	R/W	It enables to change CMD_ENC_RC_TARGET_RATE.	0x0
[7:2]	RSVD	R/W	Reserved	0x0
[1]	ENABLE_SET_SEQ_INTRA_PARAM	R/W	It enables to set SEQ_INTRA_PARAM.	0x0
[0]	ENABLE_SET_PPS_PARAM	R/W	It enables to change CMD_ENC_SEQ_PPS_PARAM.	0x0

1.2.4.2.7.3. CMD_ENC_SEQ_SRC_SIZE (0x0000011C)

A size of source picture (MANDATORY)

Table 1.290. CMD_ENC_SEQ_SRC_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_HEIGHT																SRC_WIDTH															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.291. CMD_ENC_SEQ_SRC_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	SRC_HEIGHT	R/W	A height of source picture in pixel (128 ~ 8192)	0x0
[15:0]	SRC_WIDTH	R/W	A width of source picture in pixel (256 ~ 8192)	0x0

1.2.4.2.7.4. CMD_ENC_SEQ_CUSTOM_MAP_ENDIAN (0x00000120)

Custom map endian (MANDATORY)

Table 1.292. CMD_ENC_SEQ_CUSTOM_MAP_ENDIAN Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUSTOM_MAP_ENDIAN																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.293. CMD_ENC_SEQ_CUSTOM_MAP_ENDIAN Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CUSTOM_MAP_ENDIAN	R/W	Endianness	0x0

1.2.4.2.7.5. CMD_ENC_SEQ_SPS_PARAM (0x00000124)

HEVC encoder sequence parameters (MANDATORY)

Table 1.294. CMD_ENC_SEQ_SPS_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		SVC_MODE	EN_SVC	USE_STRONG_INTRA	USE_INTRA_TRANS_SKIP		USE_SAO	USE_TMVP	USE_SCALING_LIST	USE_LONG_TERM	CHROMA_FORMAT_IDC		BIT_DEPTH			TIER_IDC		LEVEL_IDC						PROFILE_IDC							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.295. CMD_ENC_SEQ_SPS_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:30]	RSVD	R/W	Reserved	0x0
[29]	SVC_MODE	R/W	SVC mode (SVAC only) 0 : disable inter-layer prediction. (simulcast) 1 : enable inter-layer prediction in which EL picture is predicted with motion vector information from the base layer picture. This flag is valid only if EN_SVC = 1	0x0
[28]	EN_SVC	R/W	It enables SVC. (SVAC only)	0x0
[27]	USE_STRONG_INTRA	R/W	It enables strong intra smoothing. (HEVC only)	0x0
[26:25]	USE_INTRA_TRANS_SKIP	R/W	It enables transform skip for intra CU. (HEVC only)	0x0
[24]	USE_SAO	R/W	It enables sample adaptive offset. (HEVC/SVAC)	0x0
[23]	USE_TMVP	R/W	It enables temporal motion vector prediction. (HEVC only)	0x0
[22]	USE_SCALING_LIST	R/W	It enables to apply user scaling list. (HEVC/AVC)	0x0
[21]	USE_LONG_TERM	R/W	It enables to use longterm reference frame.	0x0
[20:19]	CHROMA_FORMAT_IDC	R/W	chroma format indicator 0 : 4:2:0	0x0
[18:14]	BIT_DEPTH	R/W	Bit depth (8 or 10)	0x0
[13:12]	TIER_IDC	R/W	A tier indicator (HEVC only) 0 : main 1 : high	0x0
[11:3]	LEVEL_IDC	R/W	Reserved It should be 0.	0x0
[2:0]	PROFILE_IDC	R/W	Profile indicator 0 : firmware determines a profile. HEVC 1 : main 2 : main10 3 : main still picture AVC 1: base 2: main 3: high 4: high10 SVAC 1: main 2: main10	0x0

1.2.4.2.7.6. CMD_ENC_SEQ_PPS_PARAM (0x00000128)

HEVC encoder picture parameters

Table 1.296. CMD_ENC_SEQ_PPS_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD	USE_ENTROPY_CODING_MODE	USE_TRANSFORM_8x8	Y_DC_QP_OFFSET					CR_QP_OFFSET					CB_QP_OFFSET					TC_OFFSET_DIV2				BETA_OFFSET_DIV2				DISABLE_DBK	USE_WPP	USE_WP	USE_LF_CROSS_SLICE_BOUNDARY	CONSTRAINED_INTRA_PRED	USE_LOSSLESS_CODING
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.297. CMD_ENC_SEQ_PPS_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31]	RSVD	R/W	Reserved	0x0
[30]	USE_ENTROPY_CODING_MODE	R/W	enable CABAC	0x0
[29]	USE_TRANSFORM_8x8	R/W	enable transform 8x8	0x0
[28:24]	Y_DC_QP_OFFSET	R/W	A delta quantization parameter for luma DC coefficienrts (-12 ~ 12) (SVAC only)	0x0
[23:19]	CR_QP_OFFSET	R/W	HEVC/AVC: QP offset for Cr color component SVAC: A delta quantization parameter for chroma AC coefficienrts (-12 ~ 12)	0x0
[18:14]	CB_QP_OFFSET	R/W	HEVC/AVC: QP offset for Cb color component SVAC: A delta quantization parameter for chroma DC coefficienrts (-12 ~ 12)	0x0
[13:10]	TC_OFFSET_DIV2	R/W	TcOffsetDiv2 for deblocking filter (HEVC/AVC)	0x0
[9:6]	BETA_OFFSET_DIV2	R/W	BetaOffsetDiv2 for deblocking filter (HEVC/AVC)	0x0
[5]	DISABLE_DBK	R/W	disables the in-loop deblocking filter (HEVC/AVC)	0x0
[4]	USE_WPP	R/W	It enables wave-front parallel processing (entropy coding sync enabled flag = 1 @PPS). (HEVC only) **Note** This parameter cannot be changed within the same sequence.	0x0
[3]	USE_WP	R/W	It enables to use weighted prediction. (HEVC/AVC)	0x0
[2]	USE_LF_CROSS_SLICE_BOUNDARY	R/W	It enables the use of in-loop filtering across slice boundaries. (HEVC/AVC)	0x0
[1]	CONSTRAINED_INTRA_PRED	R/W	It enables constrained intra prediction. (HEVC/AVC)	0x0
[0]	USE_LOSSLESS_CODING	R/W	It enables lossless coding. (HEVC only)	0x0

Bit	Name	Type	Function	Reset Value
			Note This parameter cannot be changed within the same sequence.	

1.2.4.2.7.7. CMD_ENC_SEQ_GOP_PARAM (0x0000012C)

GOP structure designation (MANDATORY)

Table 1.298. CMD_ENC_SEQ_GOP_PARAM Bit Assignment

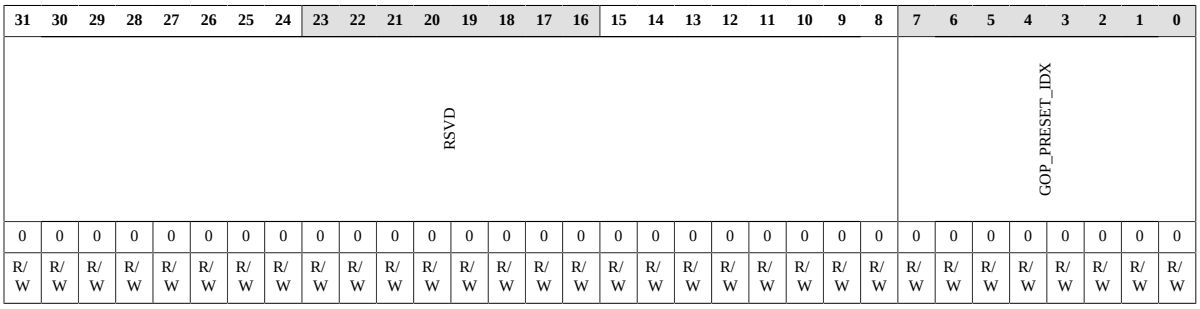


Table 1.299. CMD_ENC_SEQ_GOP_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R/W	Reserved	0x0
[7:0]	GOP_PRESET_IDX	R/W	A GOP structure option 0: Custom GOP 1 : I-I-I-I,..I (all intra, gop_size=1) 2 : I-P-P-P,… P (consecutive P, gop_size=1) 3 : I-B-B-B,…B (consecutive B, gop_size=1) 4 : I-B-P-B-P,… (gop_size=2) 5 : I-B-B-B-P,… (gop_size=4) 6 : I-P-P-P-P,… (consecutive P, gop_size=4) 7 : I-B-B-B-B,… (consecutive B, gop_size=4) 8 : I-B-B-B-B-B-B-B-B,… (random access, gop_size=8)	0x0

1.2.4.2.7.8. CMD_ENC_SEQ_INTRA_PARAM (0x00000130)

Intra picture coding parameters (MANDATORY)

Table 1.300. CMD_ENC_SEQ_INTRA_PARAM Bit Assignment

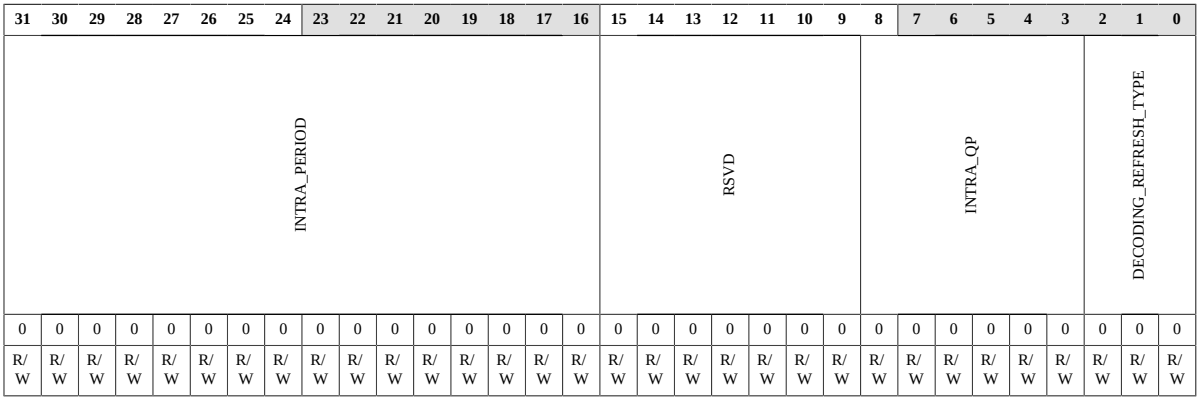


Table 1.301. CMD_ENC_SEQ_INTRA_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	INTRA_PERIOD	R/W	A period of intra picture (0 ~ 1024) 0 - implies an infinite period	0x0
[15:9]	RSVD	R/W	Reserved	0x0
[8:3]	INTRA_QP	R/W	A quantization parameter of intra picture (0 ~ 51)	0x0
[2:0]	DECODING_REFRESH_TYPE	R/W	A decoding refresh type of intra picture 0 : Non-IRAP 1 : CRA 2 : IDR **Note** This parameter cannot be changed within the same sequence.	0x0

1.2.4.2.7.9. CMD_ENC_SEQ_CONF_WIN_TOP_BOT (0x00000134)

Top and bottom size for conformance window

Table 1.302. CMD_ENC_SEQ_CONF_WIN_TOP_BOT Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONFORMACE_WINDOW_SIZE_BOTTOM																CONFORMACE_WINDOW_SIZE_TOP															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.303. CMD_ENC_SEQ_CONF_WIN_TOP_BOT Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CONFORMACE_WINDOW_SIZE_BOTTOM	R/W	A conformance bottom window size (0 ~ 8192)	0x0
[15:0]	CONFORMACE_WINDOW_SIZE_TOP	R/W	A conformance top window size (0 ~ 8192)	0x0

1.2.4.2.7.10. CMD_ENC_SEQ_CONF_WIN_LEFT_RIGHT (0x00000138)

Left and right size for conformance window

Table 1.304. CMD_ENC_SEQ_CONF_WIN_LEFT_RIGHT Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

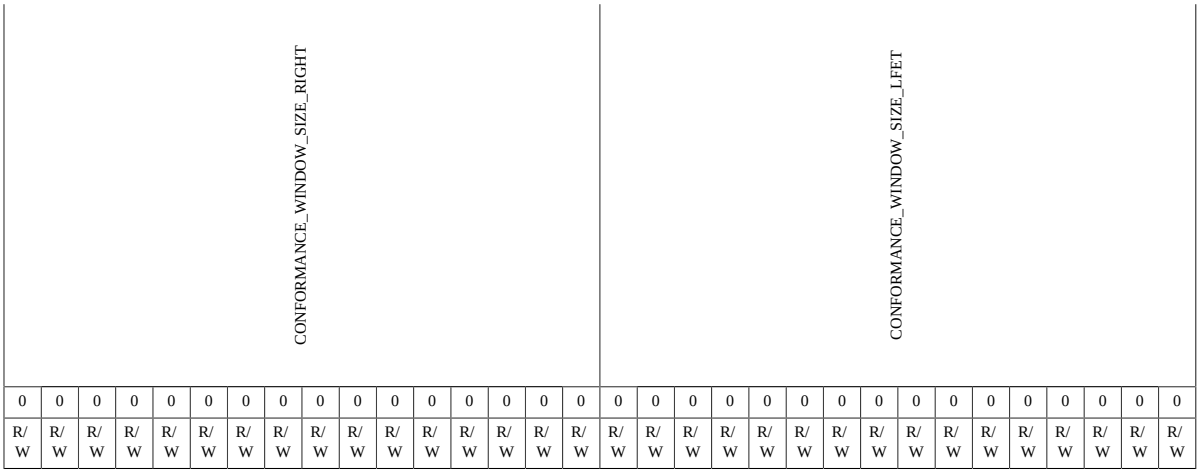


Table 1.305. CMD_ENC_SEQ_CONF_WIN_LEFT_RIGHT Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CONFORMANCE_WINDOW_SIZE_RIGHT	R/W	A conformance right window size (0 ~ 8192)	0x0
[15:0]	CONFORMANCE_WINDOW_SIZE_LFET	R/W	A conformance left window size (0 ~ 8192)	0x0

1.2.4.2.7.11. CMD_ENC_SEQ_RDO_PARAM (0x0000013C)

RDO coding options

Table 1.306. CMD_ENC_SEQ_RDO_PARAM Bit Assignment

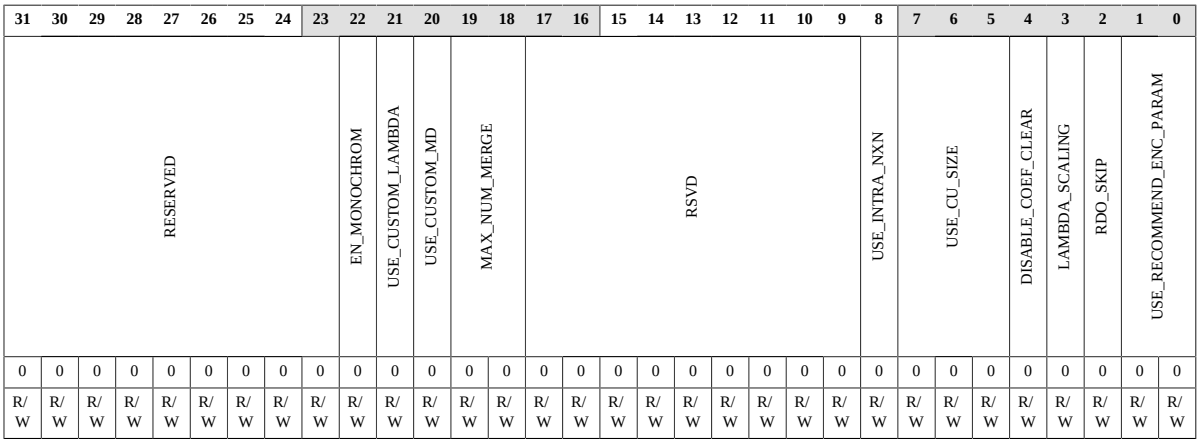


Table 1.307. CMD_ENC_SEQ_RDO_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:23]	RESERVED	R/W	Reserved	0x0
[22]	EN_MONOCHROM	R/W	It enables monochrom encoding mode. **Note** This parameter cannot be changed within the same sequence.	0x0
[21]	USE_CUSTOM_LAMBDA	R/W	It enables custom lambda table.	0x0

Bit	Name	Type	Function	Reset Value
[20]	USE_CUSTOM_MD	R/W	It enables custom mode decision.	0x0
[19:18]	MAX_NUM_MERGE	R/W	Maximum number of merge candidates (0 ~ 2)	0x0
[17:9]	RSVD	R/W	Reserved	0x0
[8]	USE_INTRA_NXN	R/W	It enables intra NxN PU.	0x0
[7:5]	USE_CU_SIZE	R/W	It enables CU size. [0] : 8x8 CU [1] : 16x16 CU [2] : 32x32 CU **Note** Host should enable all CU sizes due to hardware constraint. The RDO block determines the best CU size for CTU on its own. **Note** This parameter cannot be changed within the same sequence.	0x0
[4]	DISABLE_COEF_CLEAR	R/W	Disables the transform coefficient clearing algorithm in P or B picture 0 : All-zero coefficient block is evaluated additionally in RDO 1 : All-zero coefficient block is not evaluated in RDO	0x0
[3]	LAMBDA_SCALING	R/W	It enables lambda_scaling (AVC Only)	0x0
[2]	RDO_SKIP	R/W	It enables rdo_skip (AVC Only)	0x0
[1:0]	USE_RECOMMEND_ENC_PARAM	R/W	It uses recommended enc params. 0 : Custom 1 : Recommend enc params (slow encoding speed, highest picture quality) 2 : Boost mode (normal encoding speed, normal picture quality) 3 : Fast mode (high encoding speed, low picture quality) **Note** This parameter cannot be changed within the same sequence.	0x0

1.2.4.2.7.12. CMD_ENC_SEQ_INDEPENDENT_SLICE (0x00000140)

Slice parameters for an independent slice segment

Table 1.308. CMD_ENC_SEQ_INDEPENDENT_SLICE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARGUMENT																RESERVED													SLICE_MODE		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.309. CMD_ENC_SEQ_INDEPENDENT_SLICE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	ARGUMENT	R/W	A size of slice for the slice mode (2^16 - 1) • The number of CTU for slice mode of 1	0x0
[15:3]	RESERVED	R/W	Reserved	0x0
[2:0]	SLICE_MODE	R/W	An independent slice mode 0 - No multi-slice 1 - CTU based slice	0x0

1.2.4.2.7.13. CMD_ENC_SEQ_DEPENDENT_SLICE (0x00000144)

Slice parameters for a dependent slice segment

Table 1.310. CMD_ENC_SEQ_DEPENDENT_SLICE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARGUMENT																RESERVED													SLICE_MODE		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.311. CMD_ENC_SEQ_DEPENDENT_SLICE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	ARGUMENT	R/W	A size of slice for the slice mode (2^16 - 1) • The number of CTU for slice mode of 1 • The number of byte for slice mode of 2	0x0
[15:3]	RESERVED	R/W	Reserved	0x0
[2:0]	SLICE_MODE	R/W	A dependent slice mode 0 : No multi-slice 1 : CTU based slice 2 : Byte based slice	0x0

1.2.4.2.7.14. CMD_ENC_SEQ_INTRA_REFRESH (0x00000148)

Intra refresh mode

Table 1.312. CMD_ENC_SEQ_INTRA_REFRESH Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTRA_REFRESH_ARGUMENT																RSVD													INTRA_REFRESH_MODE		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.313. CMD_ENC_SEQ_INTRA_REFRESH Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	INTRA_REFRESH_ARGUMENT	R/W	It specifies an intra CTU refresh interval. Depending on intraRefreshMode, it can mean one of the followings: The number of consecutive CTU rows for IntraCtuRefreshMode of 1 The number of consecutive CTU columns for IntraCtuRefreshMode of 2 A step size in CTU for IntraCtuRefreshMode of 3 The number of Intra CTUs to be encoded in a picture for IntraCtuRefreshMode of 4	0x0
[15:3]	RSVD	R/W	Reserved	0x0
[2:0]	INTRA_REFRESH_MODE	R/W	An intra refresh mode 0 : No intra refresh 1 : Row 2 : Column 3 : A step size in CTU 4 : Adaptive intra refresh	0x0

1.2.4.2.7.15. CMD_ENC_SEQ_INPUT_SRC_PARAM (0x0000014C)

Source frame parameter (MANDATORY)

Table 1.314. CMD_ENC_SEQ_INPUT_SRC_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													CFRAME_SUBSAMPLE	CFRAME_TX16_C								CFRAME_TX16_Y								CFRAME_LOSSLESS	EN_CFRAME50
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.315. CMD_ENC_SEQ_INPUT_SRC_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:19]	RSVD	R	Reserved	0x0
[18]	CFRAME_SUBSAMPLE	R/W	It enables 422 to 420 chroma format conversion.	0x0
[17:10]	CFRAME_TX16_C	R/W	Target bit for each chroma 4x4 block	0x0
[9:2]	CFRAME_TX16_Y	R/W	Target bit for each luma 4x4 block	0x0
[1]	CFRAME_LOSSLESS	R/W	It enables lossless mode.	0x0
[0]	EN_CFRAME50	R/W	It enables VPU to get input source frames that are encoded with CFrame50.	0x0

1.2.4.2.7.16. CMD_ENC_RC_FRAME_RATE (0x00000150)

Table 1.316. CMD_ENC_RC_FRAME_RATE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

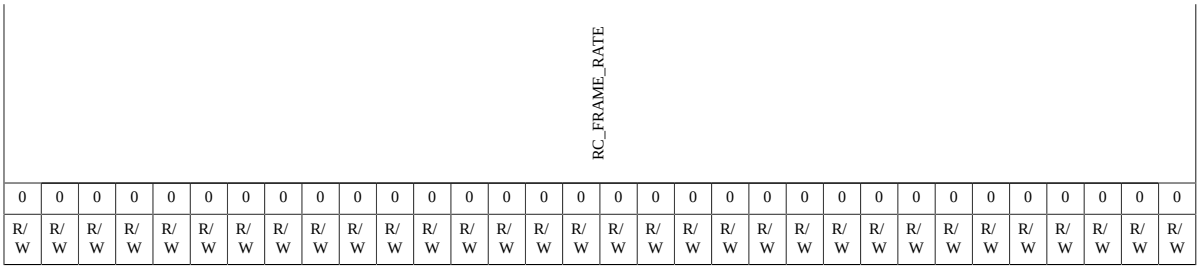


Table 1.317. CMD_ENC_RC_FRAME_RATE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RC_FRAME_RATE	R/W	A frame rate indicator (x 1024) for rate control	0x0

1.2.4.2.7.17. CMD_ENC_RC_TARGET_RATE (0x00000154)

Table 1.318. CMD_ENC_RC_TARGET_RATE Bit Assignment

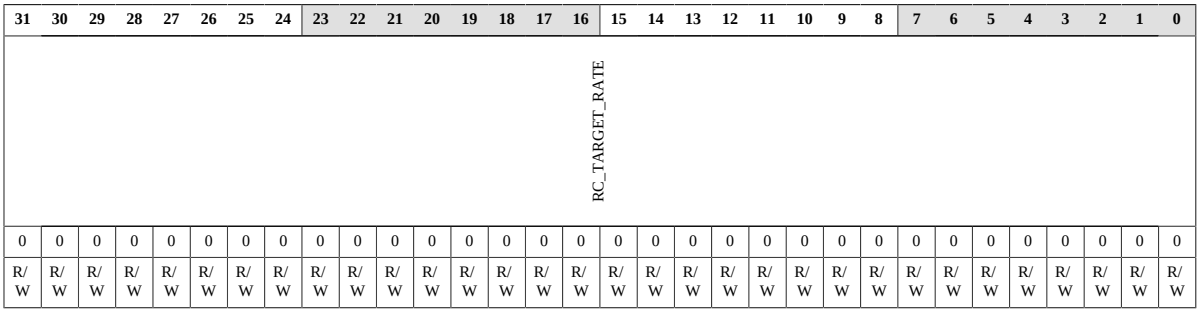


Table 1.319. CMD_ENC_RC_TARGET_RATE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RC_TARGET_RATE	R/W	A target bitrate for for rate control	0x0

1.2.4.2.7.18. CMD_ENC_RC_PARAM (0x00000158)

Table 1.320. CMD_ENC_RC_PARAM Bit Assignment

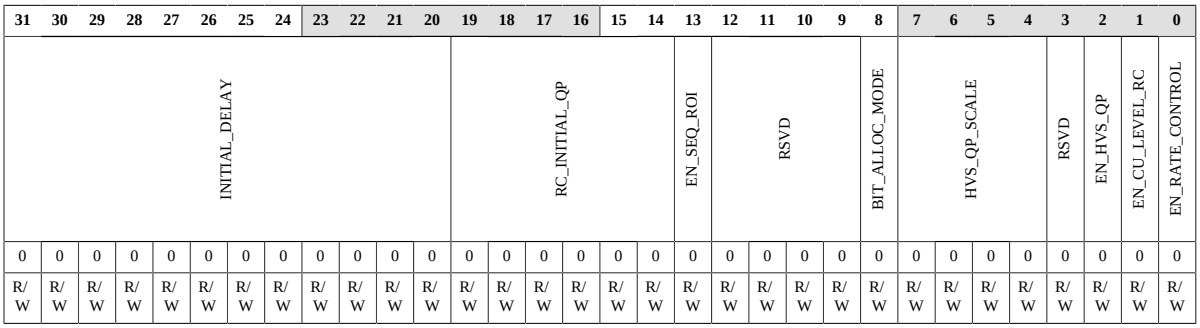


Table 1.321. CMD_ENC_RC_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:20]	INITIAL_DELAY	R/W	An inital cpb delay in msec (10 ~ 3000)	0x0
[19:14]	RC_INITIAL_QP	R/W	An initial QP	0x0

Bit	Name	Type	Function	Reset Value
			If this value is smaller than 0 or larger than 51, firmware decides the initial QP. **Note** This parameter cannot be changed within the same sequence.	
[13]	EN_SEQ_ROI	R/W	It enables ROI in sequence level. **Note** This parameter cannot be changed within the same sequence.	0x0
[12:9]	RSVD	R/W	Reserved	0x0
[8]	BIT_ALLOC_MODE	R/W	It specifies picture bits allocation mode. **Note** This parameter cannot be changed within the same sequence.	0x0
[7:4]	HVS_QP_SCALE	R/W	A QP scaling factor for CU QP adjustment with 1 of en_hvs_qp	0x0
[3]	RSVD	R/W	Reserved	0x0
[2]	EN_HVS_QP	R/W	It enables CU QP adjustment for subjective quality enhancement.	0x0
[1]	EN_CU_LEVEL_RC	R/W	It enables CU level rate control. **Note** This parameter cannot be changed within the same sequence.	0x0
[0]	EN_RATE_CONTROL	R/W	It enables rate control. **Note** This parameter cannot be changed within the same sequence.	0x0

1.2.4.2.7.19. CMD_ENC_RC_MIN_MAX_QP (0x0000015C)

Table 1.322. CMD_ENC_RC_MIN_MAX_QP Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														MAX_DELTA_QP						I_MAX_QP						I_MIN_QP					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.323. CMD_ENC_RC_MIN_MAX_QP Field Description

Bit	Name	Type	Function	Reset Value
[31:18]	RSVD	R/W	Reserved	0x0
[17:12]	MAX_DELTA_QP	R/W	A maximum delta QP for rate control (0 ~ 51)	0x0
[11:6]	I_MAX_QP	R/W	A maximum QP for intra picture (0 ~ 51)	0x0
[5:0]	I_MIN_QP	R/W	A minimum QP for intra picture (0 ~ 51)	0x0

1.2.4.2.7.20. CMD_ENC_RC_BIT_RATIO_LAYER_0_3 (0x00000160)

Table 1.324. CMD_ENC_RC_BIT_RATIO_LAYER_0_3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

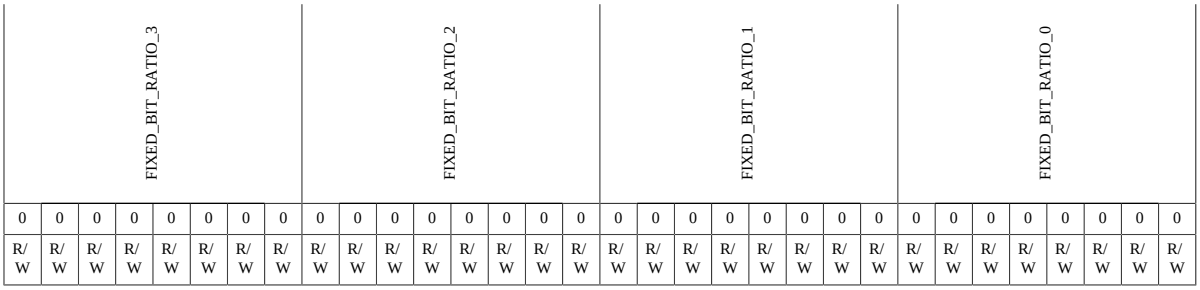


Table 1.325. CMD_ENC_RC_BIT_RATIO_LAYER_0_3 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	FIXED_BIT_RATIO_3	R/W	It specifies the ratio of 4th picture bits in encoding order in a GOP (1~255).	0x0
[23:16]	FIXED_BIT_RATIO_2	R/W	It specifies the ratio of 3rd picture bits in encoding order in a GOP (1~255).	0x0
[15:8]	FIXED_BIT_RATIO_1	R/W	It specifies the ratio of 2nd picture bits in encoding order in a GOP (1~255).	0x0
[7:0]	FIXED_BIT_RATIO_0	R/W	It specifies the ratio of 1st picture bits in encoding order in a GOP (1~255).	0x0

1.2.4.2.7.21. CMD_ENC_RC_BIT_RATIO_LAYER_4_7 (0x00000164)

Table 1.326. CMD_ENC_RC_BIT_RATIO_LAYER_4_7 Bit Assignment

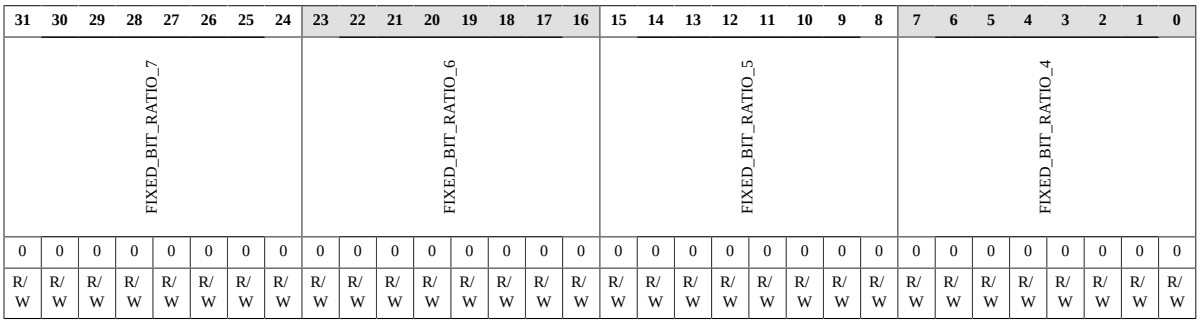
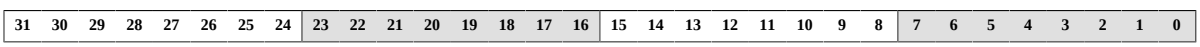


Table 1.327. CMD_ENC_RC_BIT_RATIO_LAYER_4_7 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	FIXED_BIT_RATIO_7	R/W	It specifies the ratio of 8th picture bits in encoding order in a GOP (1~255).	0x0
[23:16]	FIXED_BIT_RATIO_6	R/W	It specifies the ratio of 7th picture bits in encoding order in a GOP (1~255).	0x0
[15:8]	FIXED_BIT_RATIO_5	R/W	It specifies the ratio of 6th picture bits in encoding order in a GOP (1~255).	0x0
[7:0]	FIXED_BIT_RATIO_4	R/W	It specifies the ratio of 5th picture bits in encoding order in a GOP (1~255).	0x0

1.2.4.2.7.22. CMD_ENC_RC_INTER_MIN_MAX_QP (0x00000168)

Table 1.328. CMD_ENC_RC_INTER_MIN_MAX_QP Bit Assignment



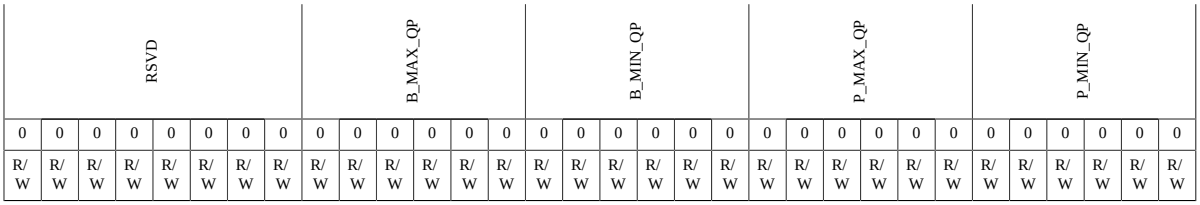


Table 1.329. CMD_ENC_RC_INTER_MIN_MAX_QP Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	RSVD	R/W	Reserved	0x0
[23:18]	B_MAX_QP	R/W	A maximum QP of B picture for rate control	0x0
[17:12]	B_MIN_QP	R/W	A minimum QP of B picture for rate control	0x0
[11:6]	P_MAX_QP	R/W	A maximum QP of P picture for rate control	0x0
[5:0]	P_MIN_QP	R/W	A minimum QP of P picture for rate control	0x0

1.2.4.2.7.23. CMD_ENC_SEQ_RESERVED (0x0000016C)

Table 1.330. CMD_ENC_SEQ_RESERVED Bit Assignment

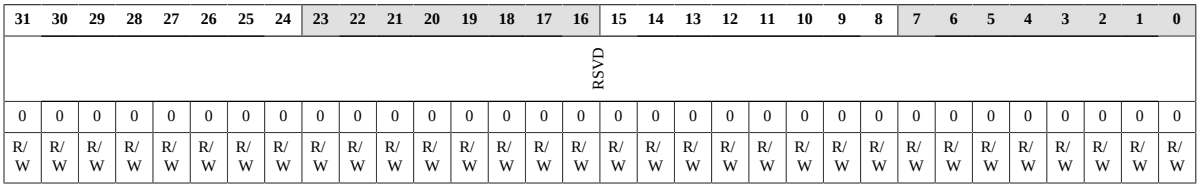


Table 1.331. CMD_ENC_SEQ_RESERVED Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R/W	Reserved	0x0

1.2.4.2.7.24. CMD_ENC_ROT_PARAM (0x00000170)

Table 1.332. CMD_ENC_ROT_PARAM Bit Assignment

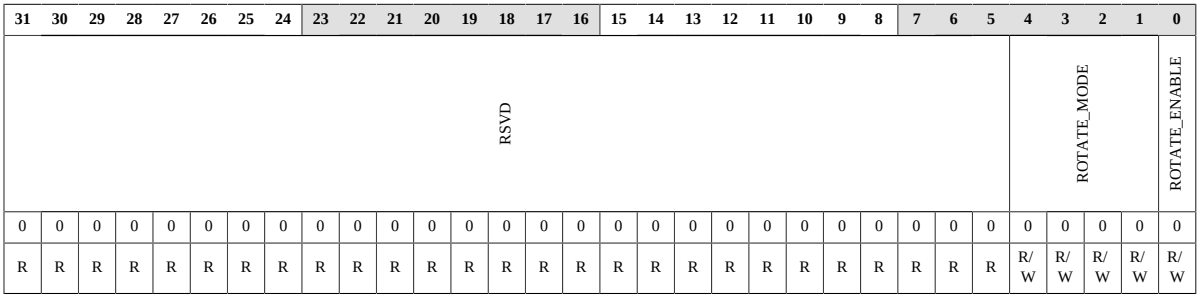


Table 1.333. CMD_ENC_ROT_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:1]	ROTATE_MODE	R/W	[3] - horizontal mirror [2] - vertical mirror [1:0] - 90 degree left rotate 1=90' 2=180'	0x0

Bit	Name	Type	Function	Reset Value
			3=270'	
[0]	ROTATE_ENABLE	R/W	It enables or disables rotation.	0x0

1.2.4.2.7.25. CMD_ENC_NUM_UNITS_IN_TICK (0x00000174)

Table 1.334. CMD_ENC_NUM_UNITS_IN_TICK Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM_UNITS_IN_TICK																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.335. CMD_ENC_NUM_UNITS_IN_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	NUM_UNITS_IN_TICK	R/W	It specifies the number of time units of a clock operating at the frequency time_scale Hz.	0x0

1.2.4.2.7.26. CMD_ENC_TIME_SCALE (0x00000178)

Table 1.336. CMD_ENC_TIME_SCALE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIME_SCALE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.337. CMD_ENC_TIME_SCALE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	TIME_SCALE	R/W	It specifies the number of time units that pass in one second.	0x0

1.2.4.2.7.27. CMD_ENC_NUM_TICKS_POC_DIFF_ONE (0x0000017C)

Table 1.338. CMD_ENC_NUM_TICKS_POC_DIFF_ONE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM_TICKS_POC_DIFF_ONE																															

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.339. CMD_ENC_NUM_TICKS_POC_DIFF_ONE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	NUM_TICKS_POC_DIFF_ONE	R/W	It specifies the number of clock ticks corresponding to a difference of picture order count values equal to 1.	0x0

1.2.4.2.7.28. CMD_ENC_CUSTOM_MD_PU04 (0x00000184)

Table 1.340. CMD_ENC_CUSTOM_MD_PU04 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU04_IM_ANG_DR								PU04_IM_DC_DR								PU04_IM_PLANAR_DR								PU04_DR							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.341. CMD_ENC_CUSTOM_MD_PU04 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PU04_IM_ANG_DR	R/W	A value which is added to rate when calculating cost (=distortion + rate) in 4x4 Angular intra prediction mode	0x0
[23:16]	PU04_IM_DC_DR	R/W	A value which is added to rate when calculating cost (=distortion + rate) in 4x4 DC intra prediction mode	0x0
[15:8]	PU04_IM_PLANAR_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 4x4 Planar intra prediction mode	0x0
[7:0]	PU04_DR	R/W	A value which is added to the total cost of 4x4 blocks	0x0

1.2.4.2.7.29. CMD_ENC_CUSTOM_MD_PU08 (0x00000188)

Table 1.342. CMD_ENC_CUSTOM_MD_PU08 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU08_IM_ANG_DR								PU08_IM_DC_DR								PU08_IM_PLANER_DR								PU08_DR							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.343. CMD_ENC_CUSTOM_MD_PU08 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PU08_IM_ANG_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 8x8 Angular intra prediction mode	0x0
[23:16]	PU08_IM_DC_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 8x8 DC intra prediction mode	0x0
[15:8]	PU08_IM_PLANER_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 8x8 Planar intra prediction mode	0x0
[7:0]	PU08_DR	R/W	A value which is added to the total cost of 8x8 blocks	0x0

1.2.4.2.7.30. CMD_ENC_CUSTOM_MD_PU16 (0x0000018C)

Table 1.344. CMD_ENC_CUSTOM_MD_PU16 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU016_IM_ANG_DR								PU016_IM_DC_DR								PU016_IM_PLANER_DR								PU016_DR							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.345. CMD_ENC_CUSTOM_MD_PU16 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PU016_IM_ANG_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 16x16 Angular intra prediction mode	0x0
[23:16]	PU016_IM_DC_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 16x16 DC intra prediction mode	0x0
[15:8]	PU016_IM_PLANER_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 16x16 Planar intra prediction mode	0x0
[7:0]	PU016_DR	R/W	A value which is added to the total cost of 16x16 blocks	0x0

1.2.4.2.7.31. CMD_ENC_CUSTOM_MD_PU32 (0x00000190)

Table 1.346. CMD_ENC_CUSTOM_MD_PU32 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU32_IM_ANG_DR								PU32_IM_DC_DR								PU32_IM_PLANER_DR								PU32_DR							

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.347. CMD_ENC_CUSTOM_MD_PU32 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	PU32_IM_ANG_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 32x32 Angular intra prediction mode	0x0
[23:16]	PU32_IM_DC_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 32x32 DC intra prediction mode	0x0
[15:8]	PU32_IM_PLANER_DR	R/W	A value which is added to rate when calculating cost(=distortion + rate) in 32x32 Planar intra prediction mode	0x0
[7:0]	PU32_DR	R/W	A value which is added to the total cost of 32x32 blocks	0x0

1.2.4.2.7.32. CMD_ENC_CUSTOM_MD_CU08 (0x00000194)

Table 1.348. CMD_ENC_CUSTOM_MD_CU08 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								CU08_MERGE_DR								CU08_INTER_DR								CU08_INTRA_DR							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.349. CMD_ENC_CUSTOM_MD_CU08 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	RSVD	R/W	Reserved	0x0
[23:16]	CU08_MERGE_DR	R/W	A value which is added to rate when calculating cost for merge CU8x8	0x0
[15:8]	CU08_INTER_DR	R/W	A value which is added to rate when calculating cost for inter CU8x8	0x0
[7:0]	CU08_INTRA_DR	R/W	A value which is added to rate when calculating cost for intra CU8x8	0x0

1.2.4.2.7.33. CMD_ENC_CUSTOM_MD_CU16 (0x00000198)

Table 1.350. CMD_ENC_CUSTOM_MD_CU16 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								CU16_MERGE_DR								CU16_INTER_DR								CU16_INTRA_DR							

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.351. CMD_ENC_CUSTOM_MD_CU16 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	RSVD	R/W	Reserved	0x0
[23:16]	CU16_MERGE_DR	R/W	A value which is added to rate when calculating cost for merge CU16x16	0x0
[15:8]	CU16_INTER_DR	R/W	A value which is added to rate when calculating cost for inter CU16x16	0x0
[7:0]	CU16_INTRA_DR	R/W	A value which is added to rate when calculating cost for intra CU16x16	0x0

1.2.4.2.7.34. CMD_ENC_CUSTOM_MD_CU32 (0x0000019C)

Table 1.352. CMD_ENC_CUSTOM_MD_CU32 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								CU32_MERGE_DR								CU32_INTER_DR								CU32_INTRA_DR							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.353. CMD_ENC_CUSTOM_MD_CU32 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	RSVD	R/W	Reserved	0x0
[23:16]	CU32_MERGE_DR	R/W	A value which is added to rate when calculating cost for merge CU32x32	0x0
[15:8]	CU32_INTER_DR	R/W	A value which is added to rate when calculating cost for inter CU32x32	0x0
[7:0]	CU32_INTRA_DR	R/W	A value which is added to rate when calculating cost for intra CU32x32	0x0

1.2.4.2.7.35. CMD_ENC_NR_PARAM (0x000001A0)

Table 1.354. CMD_ENC_NR_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				NR_NOISE_SIGMA_CR								RSVD				NR_NOISE_SIGMA_CB				NR_NOISE_SIGMA_Y								ENABLE_NR_EST_NOISE	ENABLE_NR_CR	ENABLE_NR_CB	ENABLE_NR_Y
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.355. CMD_ENC_NR_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RSVD	R/W	Reserved	0x0
[27:20]	NR_NOISE_SIGMA_CR	R/W	It specifies Cr noise standard deviation if ENABLE_NR_EST_NOISE is 0. (0 ~ 255)	0x0
[19:18]	RSVD	R	Reserved	0x0
[17:12]	NR_NOISE_SIGMA_CB	R/W	It specifies Cb noise standard deviation if ENABLE_NR_EST_NOISE is 0. (0 ~ 255)	0x0
[11:4]	NR_NOISE_SIGMA_Y	R/W	It specifies Y noise standard deviation if ENABLE_NR_EST_NOISE is 0. (0 ~ 255)	0x0
[3]	ENABLE_NR_EST_NOISE	R/W	It enables VPU to conduct noise estimation for reduction. When this is disabled, noise estimation is carried out ouside VPU.	0x0
[2]	ENABLE_NR_CR	R/W	It enables noise reduction algorithm to Cr component.	0x0
[1]	ENABLE_NR_CB	R/W	It enables noise reduction algorithm to Cb component.	0x0
[0]	ENABLE_NR_Y	R/W	It enables noise reduction algorithm to Y component.	0x0

1.2.4.2.7.36. CMD_ENC_NR_WEIGHT (0x000001A4)

Table 1.356. CMD_ENC_NR_WEIGHT Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		NR_INTER_WEIGHT_CR						NR_INTER_WEIGHT_CB				NR_INTER_WEIGHT_Y				NR_INTRA_WEIGHT_CR						NR_INTRA_WEIGHT_CB					NR_INTRA_WEIGHT_Y				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.357. CMD_ENC_NR_WEIGHT Field Description

Bit	Name	Type	Function	Reset Value
[31:30]	RESERVED	R/W	Reserved	0x0
[29:25]	NR_INTER_WEIGHT_CR	R/W	Weight to CR noise level for inter picture (0 ~ 31)	0x0
[24:20]	NR_INTER_WEIGHT_CB	R/W	Weight to CB noise level for inter picture (0 ~ 31)	0x0
[19:15]	NR_INTER_WEIGHT_Y	R/W	Weight to Y noise level for inter picture (0 ~ 31) **Note** NR_INTER_WEIGHT/4 is multiplied to the noise level that has been estimated. This weight is put for inter frame to be filtered more strongly or more weakly than just with the estimated noise level.	0x0
[14:10]	NR_INTRA_WEIGHT_CR	R/W	Weight to CR noise level for intra picture (0 ~ 31)	0x0
[9:5]	NR_INTRA_WEIGHT_CB	R/W	Weight to CB noise level for intra picture (0 ~ 31)	0x0

Bit	Name	Type	Function	Reset Value
[4:0]	NR_INTRA_WEIGHT_Y	R/W	Weight to Y noise level for intra picture (0 ~ 31) **Note** NR_INTRA_WEIGHT/4 is multiplied to the noise level that has been estimated. This weight is put for intra frame to be filtered more strongly or more weakly than just with the estimated noise level.	0x0

1.2.4.2.7.37. CMD_ENC_BG_PARAM (0x000001A8)

Table 1.358. CMD_ENC_BG_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD			BG_DELTA_QP					BG_LAMBDA_QP								BG_TH_MEAN_DIFF								BG_TH_MAX_DIFF							EN_BG_DETECT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.359. CMD_ENC_BG_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:29]	RSVD	R/W	Reserved	0x0
[28:24]	BG_DELTA_QP	R/W	It specifies the difference between the lambda QP value of background and the lambda QP value of foreground.	0x0
[23:18]	BG_LAMBDA_QP	R/W	It specifies the minimum lambda QP value to be used in the background area.	0x0
[17:10]	BG_TH_MEAN_DIFF	R/W	It specifies the threshold of mean difference that is used in s2me block. It is valid when background detection is on.	0x0
[9:1]	BG_TH_MAX_DIFF	R/W	It specifies the threshold of max difference that is used in s2me block. It is valid when background detection is on.	0x0
[0]	EN_BG_DETECT	R/W	It enables background detection. **Note** This parameter cannot be changed within the same sequence.	0x0

1.2.4.2.7.38. CMD_ENC_CUSTOM_LAMBDA_ADDR (0x000001AC)

Table 1.360. CMD_ENC_CUSTOM_LAMBDA_ADDR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUSTOM_LAMBDA_MAP_ADDR																															

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.361. CMD_ENC_CUSTOM_LAMBDA_ADDR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CUSTOM_LAMBDA_MAP_ADDR	R/W	It specifies the address of custom lambda map.	0x0

1.2.4.2.7.39. CMD_ENC_USER_SCALING_LIST_ADDR (0x000001B0)

Table 1.362. CMD_ENC_USER_SCALING_LIST_ADDR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USER_SCALING_LIST_ADDR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.363. CMD_ENC_USER_SCALING_LIST_ADDR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	USER_SCALING_LIST_ADDR	R/W	It specifies the address of user scaling list data.	0x0

1.2.4.2.8. ENC_SET_PARAM Command(GOP) Parameter Registers

There are command I/O registers where Host processor can set arguments for the ENC_SET_PARAM command. These parameters are sent to firmware only when ENC_SET_PARAM_OPTION(0x104) is set to 0x1, all of which work in the GOP setting mode.

1.2.4.2.8.1. SET_PARAM_OPTION (0x00000104)

ENC_SET_PARAM command option

Table 1.364. SET_PARAM_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																														SET_PARAM_OPTION	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W

Table 1.365. SET_PARAM_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1:0]	SET_PARAM_OPTION	R/W	0x0 : COMMON **0x1 : GOP** 0x10 : CHANGE_PARAM	0x0

1.2.4.2.8.2. CMD_ENC_CUSTOM_GOP_PARAM (0x0000011C)

Size of source pictures of custom GOP (MANDATORY)

Table 1.366. CMD_ENC_CUSTOM_GOP_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																											CUSTOM_GOP_SIZE					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W

Table 1.367. CMD_ENC_CUSTOM_GOP_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	CUSTOM_GOP_SIZE	R/W	The size of cyclic GOP (1 ~ 8) This should be the same as the number of CUSTOM_GOP_PIC_PARAM set by host	0x0

1.2.4.2.8.3. CMD_ENC_CUSTOM_GOP_PIC_PARAM_0 (0x00000120)

Parameters for the 0th picture of custom GOP (MANDATORY with CUSTOM_GOP_SIZE)

Table 1.368. CMD_ENC_CUSTOM_GOP_PIC_PARAM_0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP							POC_OFFSET				PIC_TYPE	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.369. CMD_ENC_CUSTOM_GOP_PIC_PARAM_0 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 0th picture in the custom GOP	0x0
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 0th picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 0th picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0
[11:6]	PIC_QP	R/W	A quantization parameter of 0th picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 0th picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 0th picture in the custom GOP 0 : I picture 1 : P picture 2 : B picture	0x0

1.2.4.2.8.4. CMD_ENC_CUSTOM_GOP_PIC_PARAM_1 (0x00000124)

Parameters for the 1st picture of custom GOP

Table 1.370. CMD_ENC_CUSTOM_GOP_PIC_PARAM_1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP							POC_OFFSET				PIC_TYPE	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.371. CMD_ENC_CUSTOM_GOP_PIC_PARAM_1 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 1st picture in the custom GOP	0x0

Bit	Name	Type	Function	Reset Value
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 1st picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 1st picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0
[11:6]	PIC_QP	R/W	A quantization parameter of 1st picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 1st picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 1st picture in the custom GOP 0 : I picture 1 : P picture 2 : B picture	0x0

1.2.4.2.8.5. CMD_ENC_CUSTOM_GOP_PIC_PARAM_2 (0x00000128)

Parameters for the 2nd picture of custom GOP

Table 1.372. CMD_ENC_CUSTOM_GOP_PIC_PARAM_2 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP					POC_OFFSET				PIC_TYPE		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.373. CMD_ENC_CUSTOM_GOP_PIC_PARAM_2 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 2nd picture in the custom GOP	0x0
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 2nd picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 2nd picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0
[11:6]	PIC_QP	R/W	A quantization parameter of 2nd picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 2nd picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 2nd picture in the custom GOP 0 : I picture 1 : P picture 2 : B picture	0x0

1.2.4.2.8.6. CMD_ENC_CUSTOM_GOP_PIC_PARAM_3 (0x0000012C)

Parameters for the 3rd picture of custom GOP

Table 1.374. CMD_ENC_CUSTOM_GOP_PIC_PARAM_3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP							POC_OFFSET				PIC_TYPE	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.375. CMD_ENC_CUSTOM_GOP_PIC_PARAM_3 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 3rd picture in the custom GOP	0x0
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 3rd picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 3rd picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0
[11:6]	PIC_QP	R/W	A quantization parameter of 3rd picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 3rd picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 3rd picture in the custom GOP 0 : I picture 1 : P picture 2 : B picture	0x0

1.2.4.2.8.7. CMD_ENC_CUSTOM_GOP_PIC_PARAM_4 (0x00000130)

Parameters for the 4th picture of custom GOP

Table 1.376. CMD_ENC_CUSTOM_GOP_PIC_PARAM_4 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP							POC_OFFSET				PIC_TYPE	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.377. CMD_ENC_CUSTOM_GOP_PIC_PARAM_4 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 4th picture in the custom GOP	0x0
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 4th picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 4th picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[11:6]	PIC_QP	R/W	A quantization parameter of 4th picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 4th picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 4th picture in the custom GOP 0 : I picture 1 : P picture 2 : B picture	0x0

1.2.4.2.8.8. CMD_ENC_CUSTOM_GOP_PIC_PARAM_5 (0x00000134)

Parameters for the 5th picture of custom GOP

Table 1.378. CMD_ENC_CUSTOM_GOP_PIC_PARAM_5 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP					POC_OFFSET					PIC_TYPE	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.379. CMD_ENC_CUSTOM_GOP_PIC_PARAM_5 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 5th picture in the custom GOP	0x0
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 5th picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 5th picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0
[11:6]	PIC_QP	R/W	A quantization parameter of 5th picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 5th picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 5th picture in the custom GOP 0 : I picture 1 : P picture 2 : B picture	0x0

1.2.4.2.8.9. CMD_ENC_CUSTOM_GOP_PIC_PARAM_6 (0x00000138)

Parameters for the 6th picture of custom GOP

Table 1.380. CMD_ENC_CUSTOM_GOP_PIC_PARAM_6 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP					POC_OFFSET					PIC_TYPE	

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.381. CMD_ENC_CUSTOM_GOP_PIC_PARAM_6 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 6th picture in the custom GOP	0x0
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 6th picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 6th picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0
[11:6]	PIC_QP	R/W	A quantization parameter of 6th picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 6th picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 6th picture in the custom GOP 0 : I picture 1 : P picture 2 : B picture	0x0

1.2.4.2.8.10. CMD_ENC_CUSTOM_GOP_PIC_PARAM_7 (0x0000013C)

Parameters for the 7th picture of custom GOP

Table 1.382. CMD_ENC_CUSTOM_GOP_PIC_PARAM_7 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				TEMPORAL_ID				REF0_POC					REF1_POC					NUM_REF_PIC_L0	RESERVED	PIC_QP					POC_OFFSET					PIC_TYPE	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.383. CMD_ENC_CUSTOM_GOP_PIC_PARAM_7 Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RESERVED	R/W	Reserved	0x0
[27:24]	TEMPORAL_ID	R/W	A temporal ID of 7th picture in the custom GOP	0x0
[23:19]	REF0_POC	R/W	A poc offset of reference L1 of 7th picture in the custom GOP	0x0
[18:14]	REF1_POC	R/W	A poc offset of reference L0 of 7th picture in the custom GOP	0x0
[13]	NUM_REF_PIC_L0	R/W	Flag to use multi reference picture for P picture It is valid only if PIC_TYPE is P	0x0
[12]	RESERVED	R/W	Reserved	0x0
[11:6]	PIC_QP	R/W	A quantization parameter of 7th picture in the custom GOP	0x0
[5:2]	POC_OFFSET	R/W	A poc offset of 7th picture in the custom GOP	0x0
[1:0]	PIC_TYPE	R/W	A picture type of 7th picture in the custom GOP 0 : I picture 1 : P picture	0x0

Bit	Name	Type	Function	Reset Value
			2 : B picture	

1.2.4.2.9. SET_FB Command Parameter Registers for Encoder

These are command I/O registers where Host processor can set arguments for the SET_FB command or get return values.

1.2.4.2.9.1. CMD_SET_FB_OPTION (0x00000104)

Table 1.384. CMD_SET_FB_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				SVC_LAYER_FLAG	NON_REF_FBC_WRITING	RSVD						FB_ENDIAN				RSVD											SETUP_DONE	FB_GROUP_INDICATOR	RSVD		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	RW	RW	RW

Table 1.385. CMD_SET_FB_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:28]	RSVD	R	Reserved	0x0
[27]	SVC_LAYER_FLAG	R/W	It specifies the SVC layer type for the frame buffers. (SVAC ENCODER ONLY) 0 : Base layer picture 1 : Enhance layer picture The default 0 is in non-SVC case.	0x0
[26]	NON_REF_FBC_WRITING	R/W	0 : disable FBC writing for non-reference pictures. (default) 1 : enable FBC writing for all pictures for verification.	0x0
[25:20]	RSVD	R/W	Reserved	0x0
[19:16]	FB_ENDIAN	R/W	Framebuffer endianness	0x0
[15:5]	RSVD	R	Reserved	0x0
[4]	SETUP_DONE	R/W	Setup Done Please set this flag as 1 when setup for frame buffer is done	0x0
[3]	FB_GROUP_INDICATOR	R/W	First framebuffer group indicator	0x0
[2:0]	RSVD	RW	Reserved	0x0

1.2.4.2.9.2. CMD_SET_FB_COMMON_PIC_INFO (0x00000118)

DPB information

Table 1.386. CMD_SET_FB_COMMON_PIC_INFO Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		NV21_FLAG													CBCR_INTERLEAVE	FB_STRIDE															

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.387. CMD_SET_FB_COMMON_PIC_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:30]	RSVD	R	Reserved	0x0
[29]	NV21_FLAG	R/W	This field enables NV21 format for BWB output. If this field is 0, BWB output color format is NV12. Otherwise, BWB output color format is NV21. • The value of NV21 is valid when both BWB and Cb/Cr interleave are enabled. • This field is valid only if WTL_ENABLE = 1	0x0
[28:17]	RSVD	R/W	Reserved	0x0
[16]	CBCR_INTERLEAVE	R/W	Cb/Cr interleaved 1 : set linear frame buffer output as Chroma interleaved format (semi-planar format) 0 : set linear frame buffer output as planar format (separated format) This flag is valid only if WTL_ENABLE = 1	0x0
[15:0]	FB_STRIDE	R/W	Stride for the storing alignment	0x0

1.2.4.2.9.3. CMD_SET_FB_PIC_SIZE (0x0000011C)

Decoded picture size

Table 1.388. CMD_SET_FB_PIC_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIC_WIDTH																PIC_HEIGHT															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.389. CMD_SET_FB_PIC_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	PIC_WIDTH	R/W	Picture width of DPB	0x0
[15:0]	PIC_HEIGHT	R/W	Picture height of DPB	0x0

1.2.4.2.9.4. CMD_SET_FB_NUM_FB (0x00000120)

The number of start/end frame buffer to set (max. 8 frame buffers can be set for one SET_FB command

Table 1.390. CMD_SET_FB_NUM_FB Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																			FB_START_IDX					RSVD			FB_END_IDX				

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R	R	R	R/W	R/W	R/W	R/W	R/W

Table 1.391. CMD_SET_FB_NUM_FB Field Description

Bit	Name	Type	Function	Reset Value
[31:13]	RSVD	R	Reserved	0x0
[12:8]	FB_START_IDX	R/W	The number of start frame buffer to set	0x0
[7:5]	RSVD	R	Reserved	0x0
[4:0]	FB_END_IDX	R/W	The number of end frame buffer to set	0x0

1.2.4.2.9.5. CMD_SET_FB_FBC_STRIDE (0x00000128)

Frame buffer setting for compressed frame

Table 1.392. CMD_SET_FB_FBC_STRIDE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_LUMA_STRIDE																FBC_CHROMA_STRIDE															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.393. CMD_SET_FB_FBC_STRIDE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	FBC_LUMA_STRIDE	R/W	A stride of fbc for luma component	0x0
[15:0]	FBC_CHROMA_STRIDE	R/W	A stride of fbc for chroma component	0x0

1.2.4.2.9.6. CMD_SET_FB_ADDR_SUB_SAMPLED_FB_BASE (0x0000012C)

Base address of frame buffer for sub-sampled frame

Table 1.394. CMD_SET_FB_ADDR_SUB_SAMPLED_FB_BASE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_SUB_SAMPLED_FB																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.395. CMD_SET_FB_ADDR_SUB_SAMPLED_FB_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ADDR_SUB_SAMPLED_FB	R/W	A base address of frame buffer for sub sampled decoded frames	0x0

1.2.4.2.9.7. CMD_SET_FB_SUB_SAMPLED_ONE_FB_SIZE (0x00000130)

Size of frame buffer for sub-sampled frame

Table 1.396. CMD_SET_FB_SUB_SAMPLED_ONE_FB_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUB_SAMPLED_ONE_FB_SIZE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.397. CMD_SET_FB_SUB_SAMPLED_ONE_FB_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SUB_SAMPLED_ONE_FB_SIZE	R/W	A size of one sub-sampled decoded frame	0x0

1.2.4.2.9.8. CMD_SET_FB_ADDR_LUMA_BASE0 (0x00000134)

Luma base of index0

Table 1.398. CMD_SET_FB_ADDR_LUMA_BASE0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LUMA_BASE0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.399. CMD_SET_FB_ADDR_LUMA_BASE0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE0	R/W	Luma base address of DPB index 0 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 0.	0x0

1.2.4.2.9.9. CMD_SET_FB_ADDR_CB_BASE0 (0x00000138)

Cb base of index0

Table 1.400. CMD_SET_FB_ADDR_CB_BASE0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB_BASE0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.401. CMD_SET_FB_ADDR_CB_BASE0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE0	R/W	Cb base address of DPB index 0 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 0.	0x0

1.2.4.2.9.10. CMD_SET_FB_ADDR_CR_BASE0 (0x0000013C)

Cr base of index0

Table 1.402. CMD_SET_FB_ADDR_CR_BASE0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR_BASE0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.403. CMD_SET_FB_ADDR_CR_BASE0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE0	R/W	Cr base address of DPB index 0 unless FBC is used.	0x0

1.2.4.2.9.11. CMD_SET_FB_ADDR_FBC_Y_OFFSET0 (0x0000013C)

FBC luma offset base of index0

Table 1.404. CMD_SET_FB_ADDR_FBC_Y_OFFSET0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_LUMA_OFFSET_BASE0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.405. CMD_SET_FB_ADDR_FBC_Y_OFFSET0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_LUMA_OFFSET_BASE0	R/W	Compressed Luma offset table base address of DPB index 0 when FBC is used	0x0

1.2.4.2.9.12. CMD_SET_FB_ADDR_FBC_C_OFFSET0 (0x00000140)

FBC chroma offset base of index0

Table 1.406. CMD_SET_FB_ADDR_FBC_C_OFFSET0 Bit Assignment

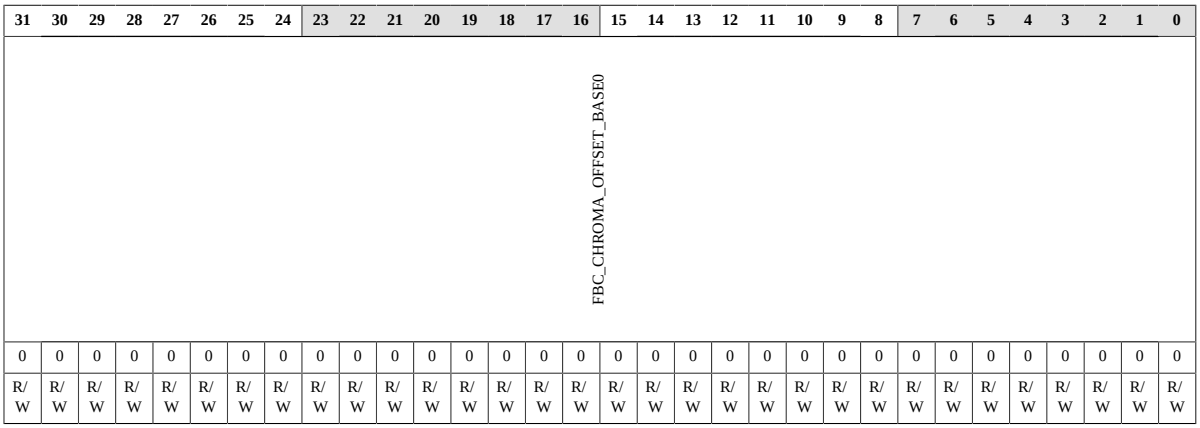


Table 1.407. CMD_SET_FB_ADDR_FBC_C_OFFSET0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE0	R/W	If FBC is used, this is chroma offset base address of DPB index 0. Otherwise, it is ignored.	0x0

1.2.4.2.9.13. CMD_SET_FB_ADDR_LUMA_BASE1 (0x00000144)

Luma base of index1

Table 1.408. CMD_SET_FB_ADDR_LUMA_BASE1 Bit Assignment

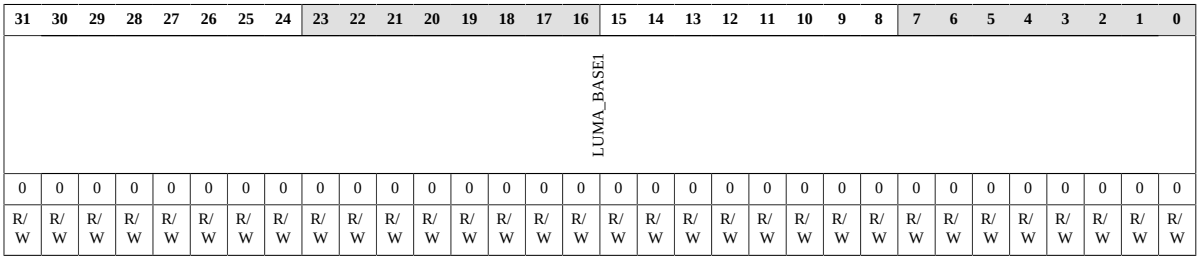


Table 1.409. CMD_SET_FB_ADDR_LUMA_BASE1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE1	R/W	Luma base address of DPB index 1 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 1.	0x0

1.2.4.2.9.14. CMD_SET_FB_ADDR_CB_BASE1 (0x00000148)

Cb base of index1

Table 1.410. CMD_SET_FB_ADDR_CB_BASE1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB_BASE1																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.411. CMD_SET_FB_ADDR_CB_BASE1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE1	R/W	Cb base address of DPB index 1 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 1.	0x0

1.2.4.2.9.15. CMD_SET_FB_ADDR_CR_BASE1 (0x0000014C)

Cr base of index1

Table 1.412. CMD_SET_FB_ADDR_CR_BASE1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR_BASE1																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.413. CMD_SET_FB_ADDR_CR_BASE1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE1	R/W	Cr base address of DPB index 1 unless FBC is used.	0x0

1.2.4.2.9.16. CMD_SET_FB_ADDR_FBC_Y_OFFSET1 (0x0000014C)

FBC luma offset base of index1

Table 1.414. CMD_SET_FB_ADDR_FBC_Y_OFFSET1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_LUMA_OFFSET_BASE1																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.415. CMD_SET_FB_ADDR_FBC_Y_OFFSET1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_LUMA_OFFSET_BASE1	R/W	Compressed Luma offset table base address of DPB index 1 when FBC is used.	0x0

1.2.4.2.9.17. CMD_SET_FB_ADDR_FBC_C_OFFSET1 (0x00000150)

FBC chroma offset base of index1

Table 1.416. CMD_SET_FB_ADDR_FBC_C_OFFSET1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_CHROMA_OFFSET_BASE1																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.417. CMD_SET_FB_ADDR_FBC_C_OFFSET1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE1	R/W	If FBC is used, this is chroma offset base address of DPB index 1. Otherwise, it is ignored.	0x0

1.2.4.2.9.18. CMD_SET_FB_ADDR_LUMA_BASE2 (0x00000154)

Luma base of index2

Table 1.418. CMD_SET_FB_ADDR_LUMA_BASE2 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LUMA_BASE2																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.419. CMD_SET_FB_ADDR_LUMA_BASE2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE2	R/W	Luma base address of DPB index 2 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 2.	0x0

1.2.4.2.9.19. CMD_SET_FB_ADDR_CB_BASE2 (0x00000158)

Cb base of index2

Table 1.420. CMD_SET_FB_ADDR_CB_BASE2 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB_BASE2																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.421. CMD_SET_FB_ADDR_CB_BASE2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE2	R/W	Cb base address of DPB index 2 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 2.	0x0

1.2.4.2.9.20. CMD_SET_FB_ADDR_CR_BASE2 (0x0000015C)

Cr base of index2

Table 1.422. CMD_SET_FB_ADDR_CR_BASE2 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR_BASE2																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.423. CMD_SET_FB_ADDR_CR_BASE2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE2	R/W	Cr base address of DPB index 2 unless FBC is used.	0x0

1.2.4.2.9.21. CMD_SET_FB_ADDR_FBC_C_OFFSET2 (0x00000160)

FBC chroma offset base of index2

Table 1.424. CMD_SET_FB_ADDR_FBC_C_OFFSET2 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_CHROMA_OFFSET_BASE2																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.425. CMD_SET_FB_ADDR_FBC_C_OFFSET2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE2	R/W	If FBC is used, this is chroma offset base address of DPB index 2. Otherwise, it is ignored.	0x0

1.2.4.2.9.22. CMD_SET_FB_ADDR_LUMA_BASE3 (0x00000164)

Luma base of index3

Table 1.426. CMD_SET_FB_ADDR_LUMA_BASE3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LUMA_BASE3																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.427. CMD_SET_FB_ADDR_LUMA_BASE3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE3	R/W	Luma base address of DPB index 3 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 3.	0x0

1.2.4.2.9.23. CMD_SET_FB_ADDR_CB_BASE3 (0x00000168)

Cb base of index3

Table 1.428. CMD_SET_FB_ADDR_CB_BASE3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB_BASE_FBC_CBCR_BASE3																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.429. CMD_SET_FB_ADDR_CB_BASE3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE_FBC_CBCR_BASE3	R/W	Cb base address of DPB index 3 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 3.	0x0

1.2.4.2.9.24. CMD_SET_FB_ADDR_CR_BASE3 (0x0000016C)

Cr base of index3

Table 1.430. CMD_SET_FB_ADDR_CR_BASE3 Bit Assignment

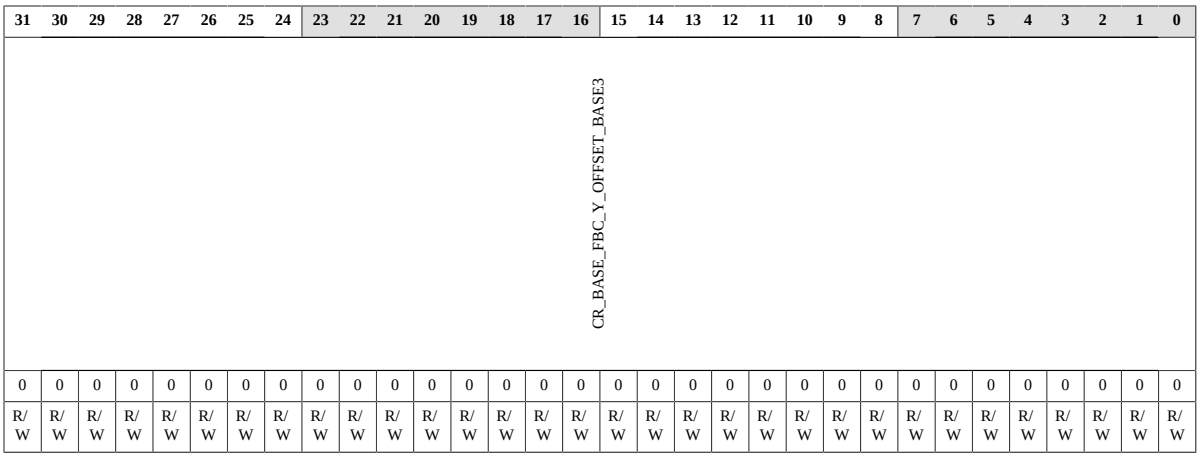


Table 1.431. CMD_SET_FB_ADDR_CR_BASE3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE3	R/W	Cr base address of DPB index 3 unless FBC is used. If FBC is used, this is compressed Luma offset table base address of DPB index 3.	0x0

1.2.4.2.9.25. CMD_SET_FB_ADDR_FBC_Y_OFFSET3 (0x0000016C)

FBC luma offset base of index3

Table 1.432. CMD_SET_FB_ADDR_FBC_Y_OFFSET3 Bit Assignment

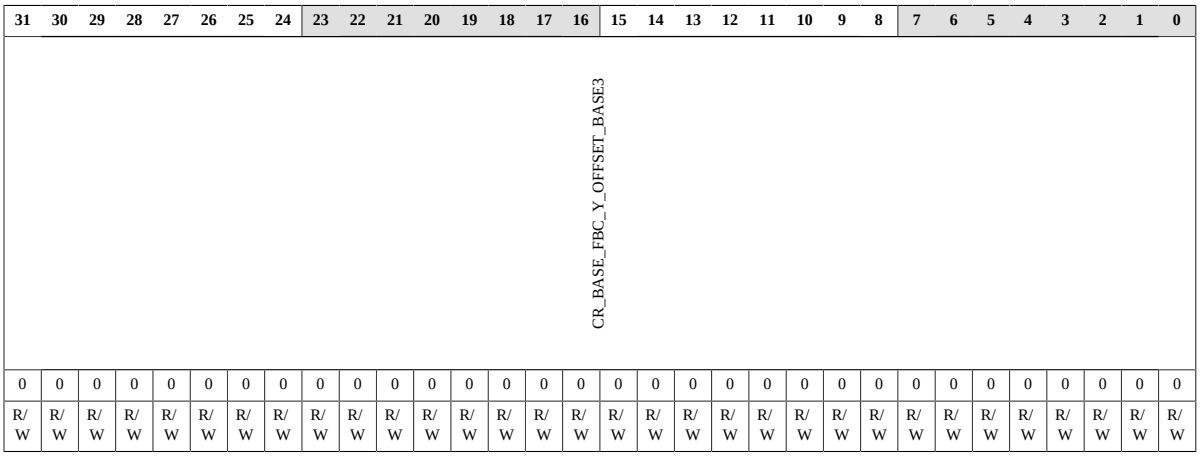


Table 1.433. CMD_SET_FB_ADDR_FBC_Y_OFFSET3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE3	R/W	Cr base address of DPB index 3 unless FBC is used.	0x0

1.2.4.2.9.26. CMD_SET_FB_ADDR_FBC_C_OFFSET3 (0x00000170)

FBC chroma offset base of index3

Table 1.434. CMD_SET_FB_ADDR_FBC_C_OFFSET3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_CHROMA_OFFSET_BASE3																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.435. CMD_SET_FB_ADDR_FBC_C_OFFSET3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE3	R/W	If FBC is used, this is chroma offset base address of DPB index 3. Otherwise, it is ignored.	0x0

1.2.4.2.9.27. CMD_SET_FB_ADDR_LUMA_BASE4 (0x00000174)

Luma base of index4

Table 1.436. CMD_SET_FB_ADDR_LUMA_BASE4 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LUMA_BASE4																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.437. CMD_SET_FB_ADDR_LUMA_BASE4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE4	R/W	Luma base address of DPB index 4 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 4.	0x0

1.2.4.2.9.28. CMD_SET_FB_ADDR_CB_BASE4 (0x00000178)

Cb base of index4

Table 1.438. CMD_SET_FB_ADDR_CB_BASE4 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

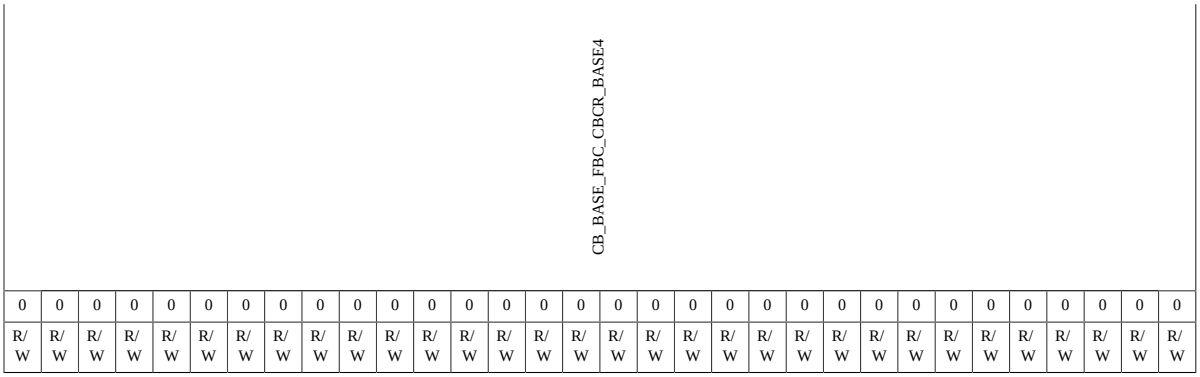


Table 1.439. CMD_SET_FB_ADDR_CB_BASE4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE_FBC_CBCR_BASE4	R/W	Cb base address of DPB index 4 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 4.	0x0

1.2.4.2.9.29. CMD_SET_FB_ADDR_CR_BASE4 (0x0000017C)

Cr base of index4

Table 1.440. CMD_SET_FB_ADDR_CR_BASE4 Bit Assignment

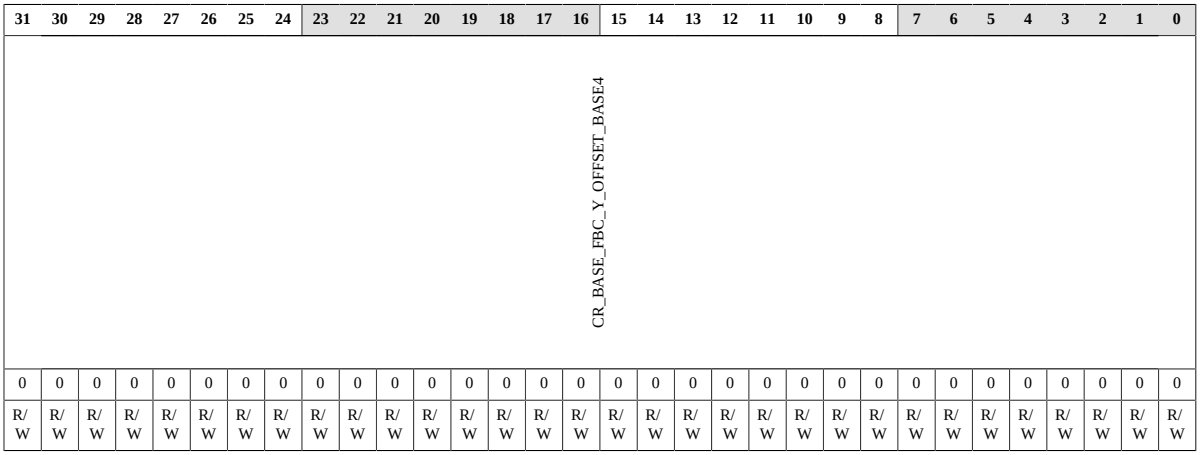


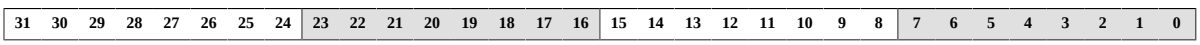
Table 1.441. CMD_SET_FB_ADDR_CR_BASE4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE4	R/W	Cr base address of DPB index 4 unless FBC is used. If FBC is used, this is compressed Luma offset table base address of DPB index 4.	0x0

1.2.4.2.9.30. CMD_SET_FB_ADDR_FBC_Y_OFFSET4 (0x0000017C)

Cr base of index4

Table 1.442. CMD_SET_FB_ADDR_FBC_Y_OFFSET4 Bit Assignment



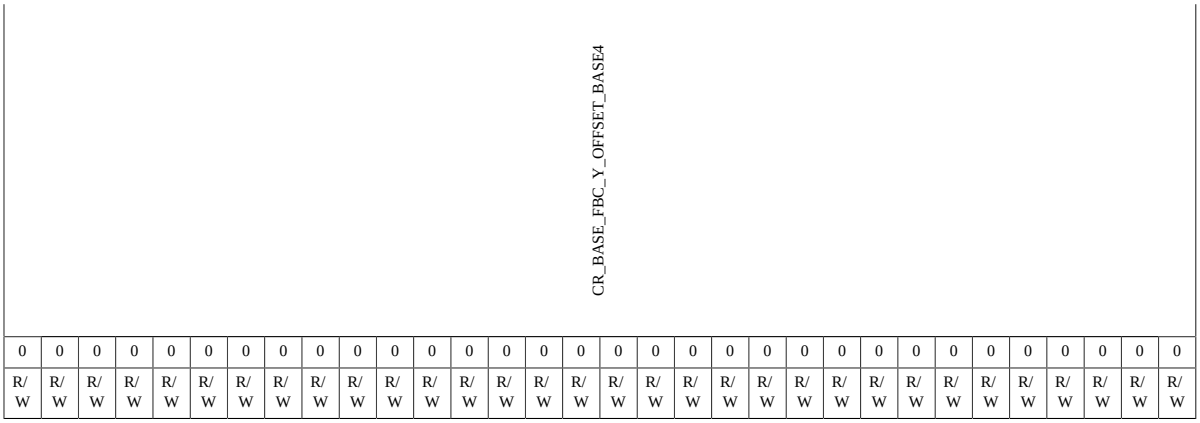


Table 1.443. CMD_SET_FB_ADDR_FBC_Y_OFFSET4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE4	R/W	Cr base address of DPB index 4 unless FBC is used. If FBC is used, this is compressed Luma offset table base address of DPB index 4.	0x0

1.2.4.2.9.31. CMD_SET_FB_ADDR_FBC_C_OFFSET4 (0x00000180)

FBC chroma offset base of index4

Table 1.444. CMD_SET_FB_ADDR_FBC_C_OFFSET4 Bit Assignment

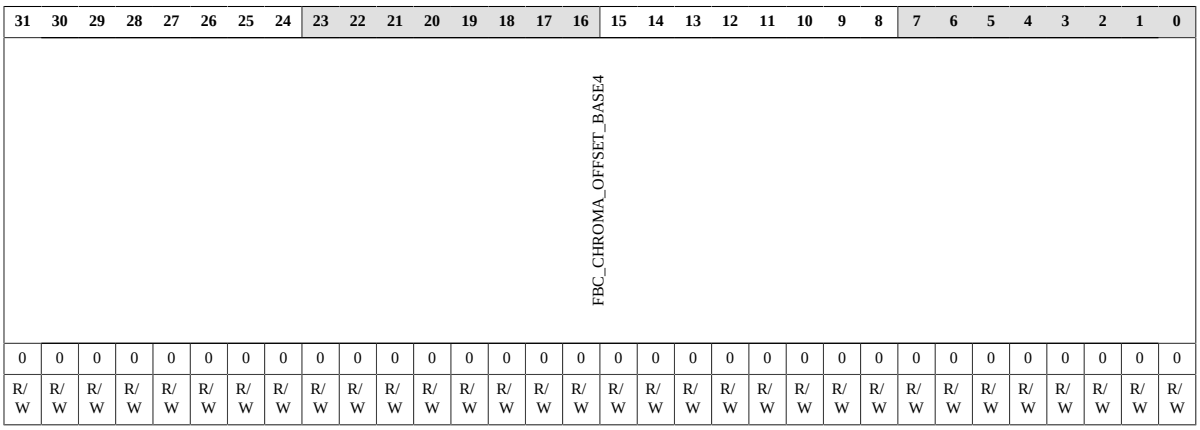


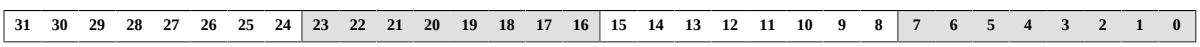
Table 1.445. CMD_SET_FB_ADDR_FBC_C_OFFSET4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE4	R/W	If FBC is used, this is chroma offset base address of DPB index 4. Otherwise, it is ignored.	0x0

1.2.4.2.9.32. CMD_SET_FB_ADDR_LUMA_BASE5 (0x00000184)

Luma base of index5

Table 1.446. CMD_SET_FB_ADDR_LUMA_BASE5 Bit Assignment



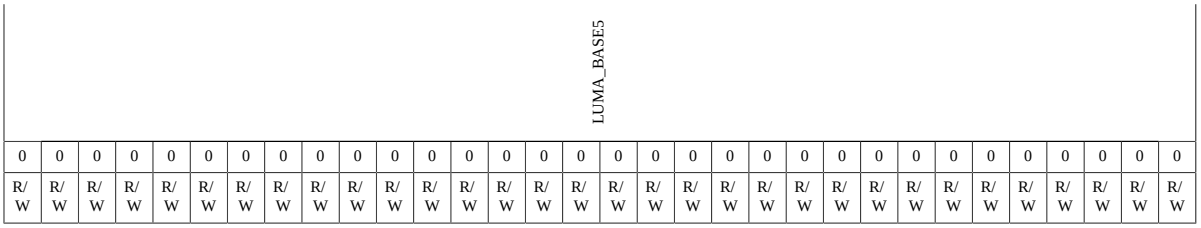


Table 1.447. CMD_SET_FB_ADDR_LUMA_BASE5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE5	R/W	Luma base address of DPB index 5 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 5.	0x0

1.2.4.2.9.33. CMD_SET_FB_ADDR_CB_BASE5 (0x00000188)

Cb base of index5

Table 1.448. CMD_SET_FB_ADDR_CB_BASE5 Bit Assignment

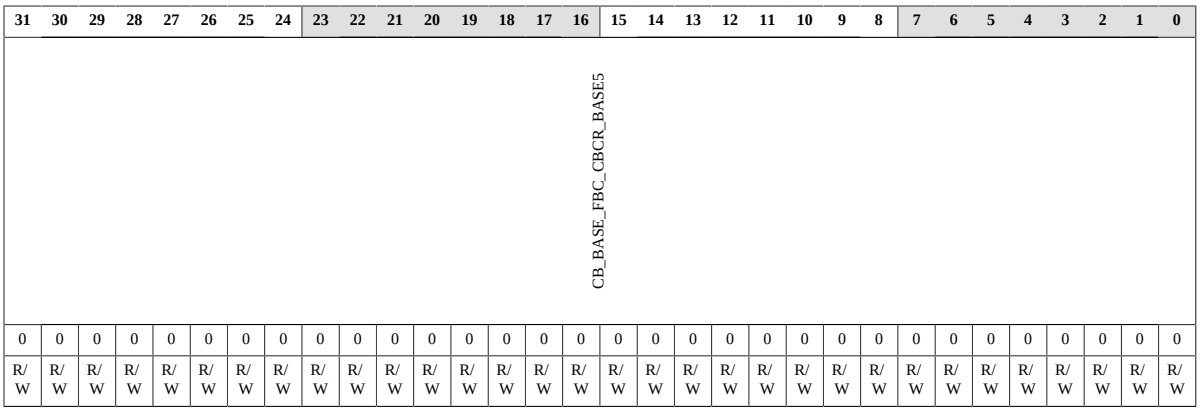


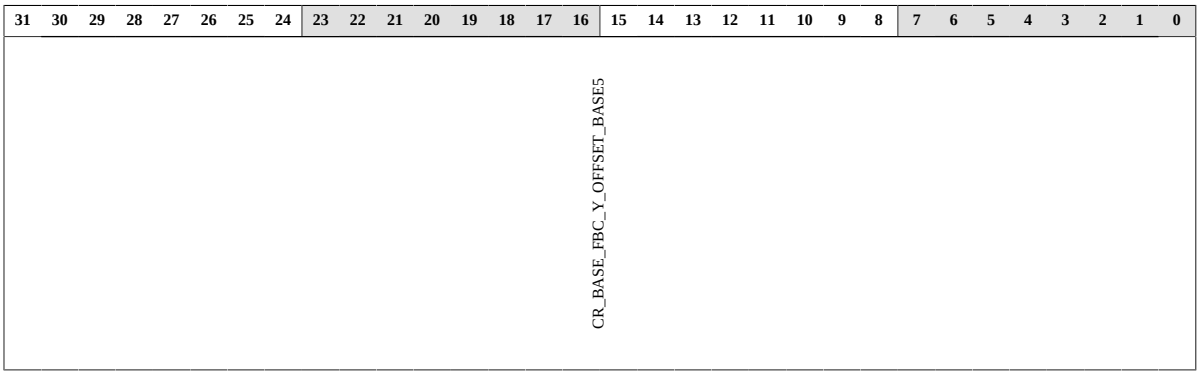
Table 1.449. CMD_SET_FB_ADDR_CB_BASE5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE_FBC_CBCR_BASE5	R/W	Cb base address of DPB index 5 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 5.	0x0

1.2.4.2.9.34. CMD_SET_FB_ADDR_CR_BASE5 (0x0000018C)

Cr base of index5

Table 1.450. CMD_SET_FB_ADDR_CR_BASE5 Bit Assignment



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.451. CMD_SET_FB_ADDR_CR_BASE5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE5	R/W	Cr base address of DPB index 5 unless FBC is used. If FBC is used, this is compressed Luma offset table base address of DPB index 5.	0x0

1.2.4.2.9.35. CMD_SET_FB_ADDR_FBC_Y_OFFSET5 (0x0000018C)

FBC luma offset base of index5

Table 1.452. CMD_SET_FB_ADDR_FBC_Y_OFFSET5 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR_BASE_FBC_Y_OFFSET_BASE5																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.453. CMD_SET_FB_ADDR_FBC_Y_OFFSET5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE5	R/W	Cr base address of DPB index 5 unless FBC is used.	0x0

1.2.4.2.9.36. CMD_SET_FB_ADDR_FBC_C_OFFSET5 (0x00000190)

FBC chroma offset base of index5

Table 1.454. CMD_SET_FB_ADDR_FBC_C_OFFSET5 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_CHROMA_OFFSET_BASE5																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.455. CMD_SET_FB_ADDR_FBC_C_OFFSET5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE5	R/W	If FBC is used, this is chroma offset base address of DPB index 5. Otherwise, it is ignored.	0x0

1.2.4.2.9.37. CMD_SET_FB_ADDR_LUMA_BASE6 (0x00000194)

Luma base of index6

Table 1.456. CMD_SET_FB_ADDR_LUMA_BASE6 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LUMA_BASE6																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.457. CMD_SET_FB_ADDR_LUMA_BASE6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE6	R/W	Luma base address of DPB index 6 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 6.	0x0

1.2.4.2.9.38. CMD_SET_FB_ADDR_CB_BASE6 (0x00000198)

Cb base of index6

Table 1.458. CMD_SET_FB_ADDR_CB_BASE6 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB_BASE_FBC_CBCR_BASE6																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.459. CMD_SET_FB_ADDR_CB_BASE6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE_FBC_CBCR_BASE6	R/W	Cb base address of DPB index 6 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 6.	0x0

1.2.4.2.9.39. CMD_SET_FB_ADDR_CR_BASE6 (0x0000019C)

Cr base of index6

Table 1.460. CMD_SET_FB_ADDR_CR_BASE6 Bit Assignment

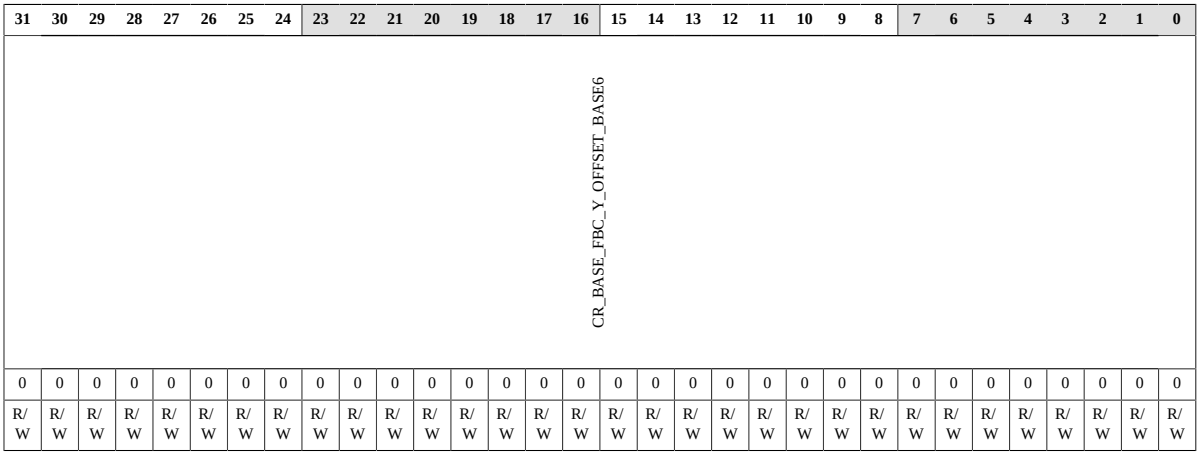


Table 1.461. CMD_SET_FB_ADDR_CR_BASE6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE6	R/W	Cr base address of DPB index 6 unless FBC is used.	0x0

1.2.4.2.9.40. CMD_SET_FB_ADDR_FBC_Y_OFFSET6 (0x0000019C)

FBC luma offset base of index6

Table 1.462. CMD_SET_FB_ADDR_FBC_Y_OFFSET6 Bit Assignment

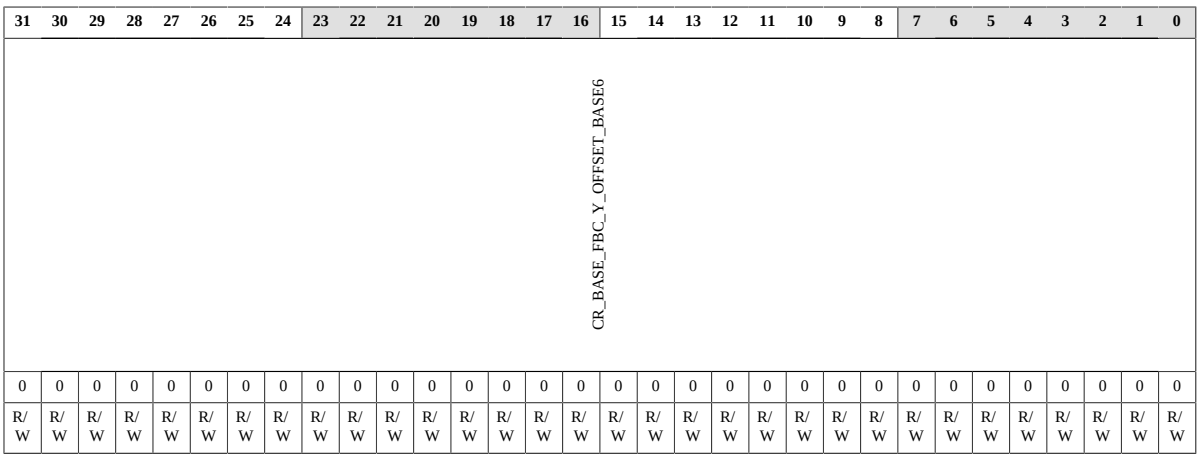


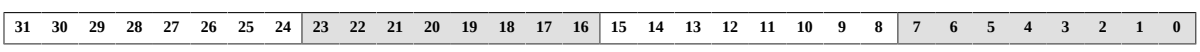
Table 1.463. CMD_SET_FB_ADDR_FBC_Y_OFFSET6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE6	R/W	Compressed Luma offset table base address of DPB index 6 when FBC is used.	0x0

1.2.4.2.9.41. CMD_SET_FB_ADDR_FBC_C_OFFSET6 (0x000001A0)

FBC chroma offset base of index6

Table 1.464. CMD_SET_FB_ADDR_FBC_C_OFFSET6 Bit Assignment



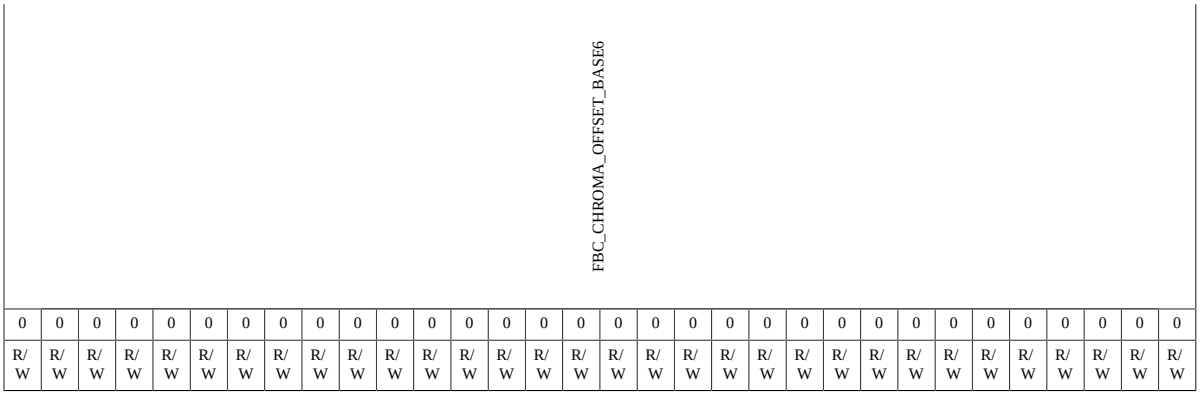


Table 1.465. CMD_SET_FB_ADDR_FBC_C_OFFSET6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE6	R/W	If FBC is used, this is chroma offset base address of DPB index 6. Otherwise, it is ignored.	0x0

1.2.4.2.9.42. CMD_SET_FB_ADDR_LUMA_BASE7 (0x000001A4)

Luma base of index7

Table 1.466. CMD_SET_FB_ADDR_LUMA_BASE7 Bit Assignment

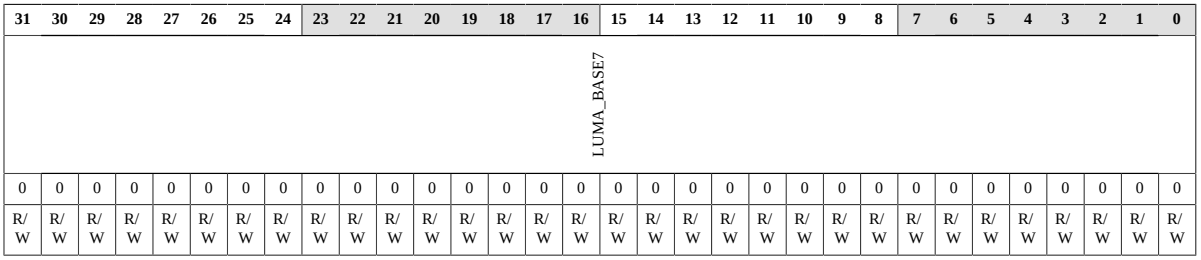


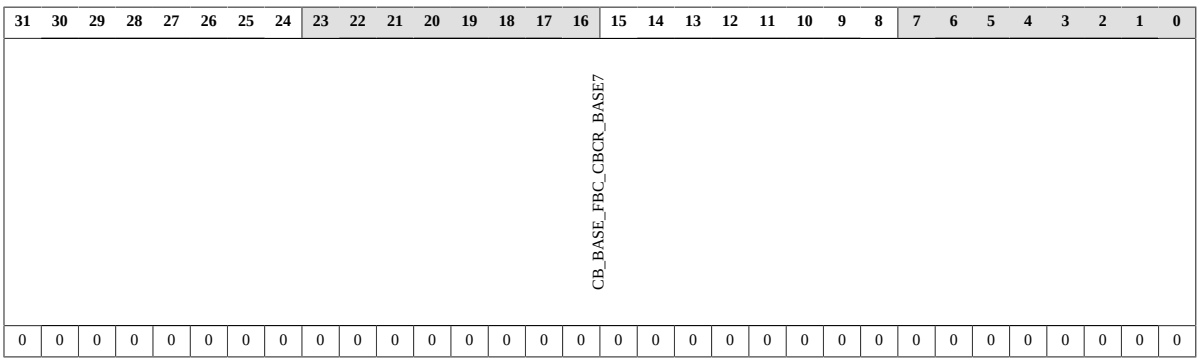
Table 1.467. CMD_SET_FB_ADDR_LUMA_BASE7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE7	R/W	Luma base address of DPB index 7 unless FBC is used. If FBC is used, this is compressed Luma base address of DPB index 7.	0x0

1.2.4.2.9.43. CMD_SET_FB_ADDR_CB_BASE7 (0x000001A8)

Cb base of index7

Table 1.468. CMD_SET_FB_ADDR_CB_BASE7 Bit Assignment



R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.469. CMD_SET_FB_ADDR_CB_BASE7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE_FBC_CBCR_BASE7	R/W	Cb base address of DPB index 7 unless FBC is used. If FBC is used, this is compressed Cb and Cr base address of DPB index 7.	0x0

1.2.4.2.9.44. CMD_SET_FB_ADDR_CR_BASE7 (0x000001AC)

Cr base of index7

Table 1.470. CMD_SET_FB_ADDR_CR_BASE7 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR_BASE_FBC_Y_OFFSET_BASE7																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.471. CMD_SET_FB_ADDR_CR_BASE7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE7	R/W	Cr base address of DPB index 7 unless FBC is used. If FBC is used, this is compressed Luma offset table base address of DPB index 7.	0x0

1.2.4.2.9.45. CMD_SET_FB_ADDR_FBC_Y_OFFSET7 (0x000001AC)

FBC luma offset base of index7

Table 1.472. CMD_SET_FB_ADDR_FBC_Y_OFFSET7 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR_BASE_FBC_Y_OFFSET_BASE7																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.473. CMD_SET_FB_ADDR_FBC_Y_OFFSET7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE_FBC_Y_OFFSET_BASE7	R/W	Compressed Luma offset table base address of DPB index 7 when FBC is used.	0x0

1.2.4.2.9.46. CMD_SET_FB_ADDR_FBC_C_OFFSET7 (0x000001B0)

FBC chroma offset base of index7

Table 1.474. CMD_SET_FB_ADDR_FBC_C_OFFSET7 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBC_CHROMA_OFFSET_BASE7																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.475. CMD_SET_FB_ADDR_FBC_C_OFFSET7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE7	R/W	If FBC is used, this is chroma offset base address of DPB index 7. Otherwise, it is ignored.	0x0

1.2.4.2.9.47. CMD_SET_FB_ADDR_MV_COL0 (0x000001B4)

Colocated mv buffer base of index 0

Table 1.476. CMD_SET_FB_ADDR_MV_COL0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COL_MV_BUF_BASE0																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.477. CMD_SET_FB_ADDR_MV_COL0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE0	R/W	HEVC : base address of colocated motion vecotors buffer of DPB index 0	0x0

Bit	Name	Type	Function	Reset Value
			SVAC : base address of colocated motion vecotors buffer for RDO process	

1.2.4.2.9.48. CMD_SET_FB_ADDR_MV_COL1 (0x000001B8)

Colocated mv buffer base of index 1

Table 1.478. CMD_SET_FB_ADDR_MV_COL1 Bit Assignment

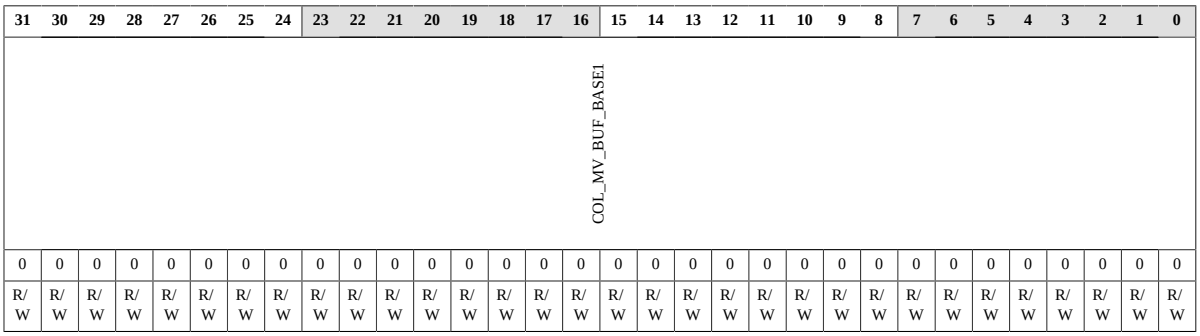


Table 1.479. CMD_SET_FB_ADDR_MV_COL1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE1	R/W	HEVC : base address of colocated motion vecotors buffer of DPB index 1 SVAC : base address of colocated motion vecotors buffer for MVP process	0x0

1.2.4.2.9.49. CMD_SET_FB_ADDR_MV_COL2 (0x000001BC)

Colocated mv buffer base of index 2

Table 1.480. CMD_SET_FB_ADDR_MV_COL2 Bit Assignment

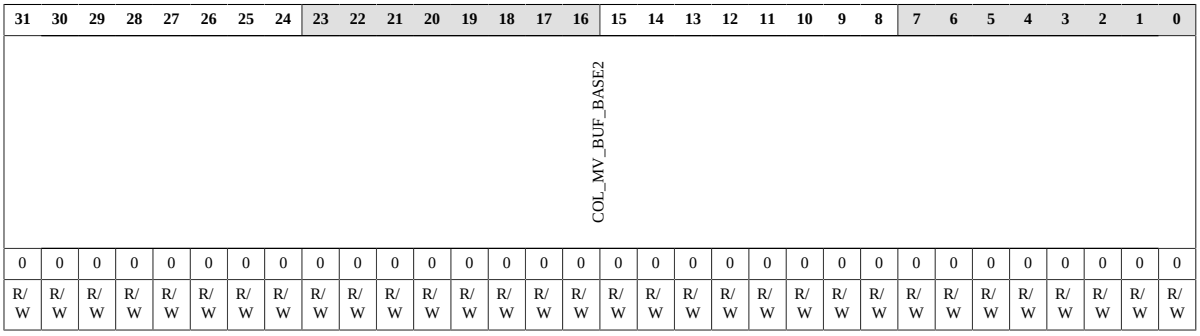


Table 1.481. CMD_SET_FB_ADDR_MV_COL2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE2	R/W	HEVC : base address of colocated motion vecotors buffer of DPB index 2 SVAC : not used	0x0

1.2.4.2.9.50. CMD_SET_FB_ADDR_MV_COL3 (0x000001C0)

Colocated mv buffer base of index 3

Table 1.482. CMD_SET_FB_ADDR_MV_COL3 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COL_MV_BUF_BASE3																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.483. CMD_SET_FB_ADDR_MV_COL3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE3	R/W	HEVC : base address of colocated motion vecotors buffer of DPB index 3 SVAC : not used	0x0

1.2.4.2.9.51. CMD_SET_FB_ADDR_MV_COL4 (0x000001C4)

Colocated mv buffer base of index 4

Table 1.484. CMD_SET_FB_ADDR_MV_COL4 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COL_MV_BUF_BASE4																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.485. CMD_SET_FB_ADDR_MV_COL4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE4	R/W	HEVC : base address of colocated motion vecotors buffer of DPB index 4 SVAC : not used	0x0

1.2.4.2.9.52. CMD_SET_FB_ADDR_MV_COL5 (0x000001C8)

Colocated mv buffer base of index 5

Table 1.486. CMD_SET_FB_ADDR_MV_COL5 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COL_MV_BUF_BASE5																															

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.487. CMD_SET_FB_ADDR_MV_COL5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE5	R/W	HEVC : base address of colocated motion vecotors buffer of DPB index 5 SVAC : not used	0x0

1.2.4.2.9.53. CMD_SET_FB_ADDR_MV_COL6 (0x000001CC)

Colocated mv buffer base of index 6

Table 1.488. CMD_SET_FB_ADDR_MV_COL6 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COL_MV_BUF_BASE6																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.489. CMD_SET_FB_ADDR_MV_COL6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE6	R/W	HEVC : Base address of colocated motion vecotors buffer of DPB index 6 SVAC : not used	0x0

1.2.4.2.9.54. CMD_SET_FB_ADDR_MV_COL7 (0x000001D0)

Colocated mv buffer base of index 7

Table 1.490. CMD_SET_FB_ADDR_MV_COL7 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COL_MV_BUF_BASE7																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.491. CMD_SET_FB_ADDR_MV_COL7 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE7	R/W	HEVC : Base address of colocated motion vecotors buffer of DPB index 7	0x0

Bit	Name	Type	Function	Reset Value
			SVAC : not used	

1.2.4.2.10. ENC_PIC Command Parameter Registers

These are command I/O registers where Host processor can set arguments for the ENC_PIC command.

1.2.4.2.10.1. CMD_BS_START (0x00000118)

Bitstream buffer start address

Table 1.492. CMD_BS_START Bit Assignment

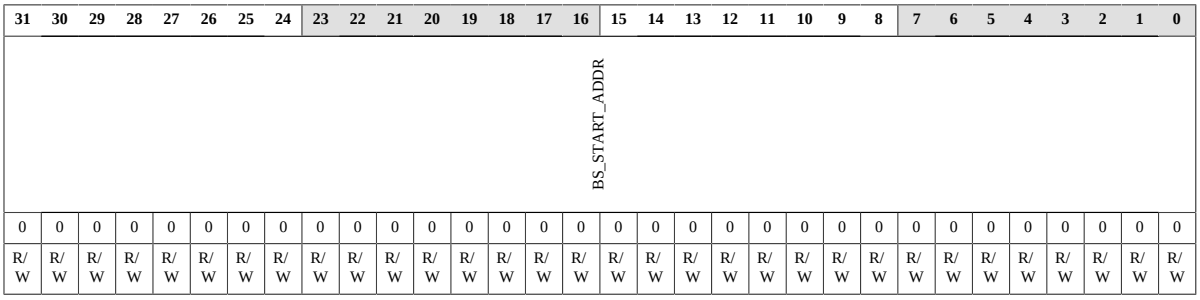


Table 1.493. CMD_BS_START Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_START_ADDR	R/W	Start address of bitstream buffer (in line buffer mode only) It should be aligned in bus width (128bit/16 byte), but we highly recommend to be aligned in 4KB	0x0

1.2.4.2.10.2. CMD_BS_SIZE (0x0000011C)

Bitstream buffer size

Table 1.494. CMD_BS_SIZE Bit Assignment

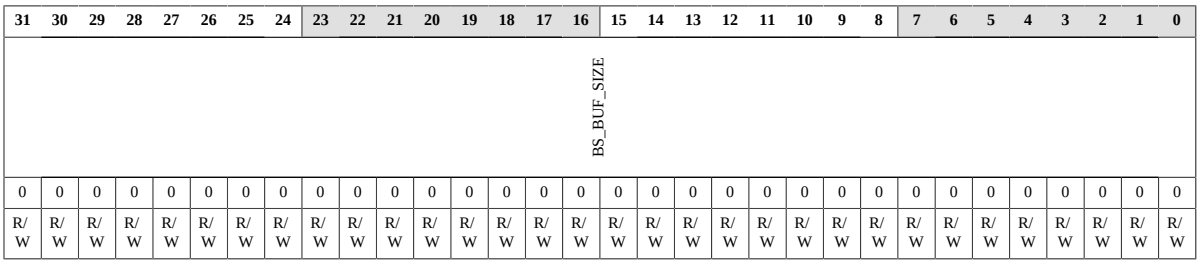


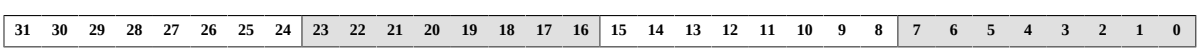
Table 1.495. CMD_BS_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_BUF_SIZE	R/W	Size of bistream buffer It should be aligned in bus width (128bit/16 byte), but we highly recommend to be aligned in 4KB	0x0

1.2.4.2.10.3. CMD_BS_OPTIONS (0x00000120)

Bistream buffer option

Table 1.496. CMD_BS_OPTIONS Bit Assignment



RSVD																							VCPU_HOST_LOWLATENCY_EN	VCORE_VCPU_LOWLATENCY_EN	LINE_BUF_INTERRUPT_EN	RSVD		BS_ENDIAN			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.497. CMD_BS_OPTIONS Field Description

Bit	Name	Type	Function	Reset Value
[31:9]	RSVD	R	Reserved	0x0
[8]	VCPU_HOST_LOWLATENCY_EN	R/W	It enables low latency mode between V-CPU and Host processor. If this is 1, V-CPU sends HOST an interrupt that a slice unit of bitstream is generated and ready to be transmitted. **Note** Low latency feature can be enabled only when the hardware supported NOTE: In case of SVAC ENC, VPU doesn't support low latency modes	0x0
[7]	VCORE_VCPU_LOWLATENCY_EN	R/W	It enables low latency mode between V-CORE and V-CPU. If this is 1, V-CORE sends V-CPU an interrupt that a slice of VLC stream is generated and ready to be transmitted to V-CPU for early start of entropy coding. **Note** Low latency feature can be enabled only when the hardware supported NOTE: In case of SVAC ENC, VPU doesn't support low latency modes	0x0
[6]	LINE_BUF_INTERRUPT_EN	R/W	TBD	0x0
[5:4]	RSVD	R/W	Reserved	0x0
[3:0]	BS_ENDIAN	R/W	Endianness of bitstream buffer	0x0

1.2.4.2.10.4. USE_SEC_AXI (0x00000124)

Secondary AXI usage option

Table 1.498. USE_SEC_AXI Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

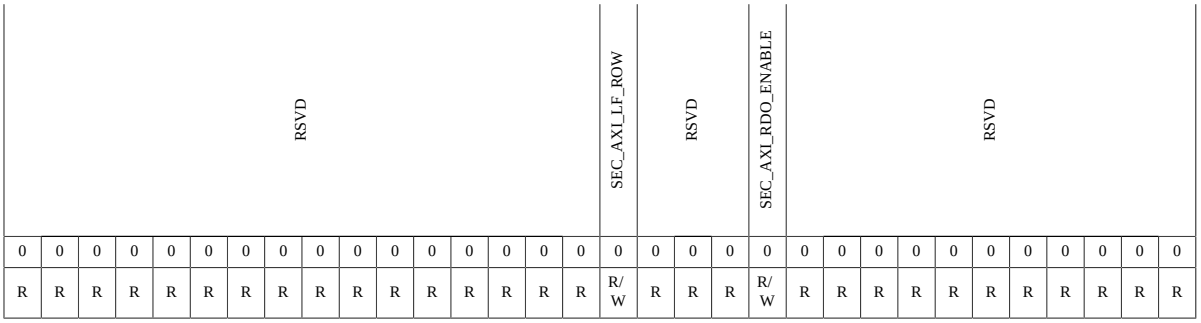


Table 1.499. USE_SEC_AXI Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15]	SEC_AXI_LF_ROW	R/W	Use 2nd AXI temp buffer for Read channel of LF row buffer.	0x0
[14:12]	RSVD	R	Reserved	0x0
[11]	SEC_AXI_RDO_ENABLE	R/W	Use 2nd AXI temp buffer for Read channel of RDO row buffer.	0x0
[10:0]	RSVD	R	Reserved	0x0

1.2.4.2.10.5. CMD_ENC_REPORT_PARAM (0x00000128)

Report parameters

Table 1.500. CMD_ENC_REPORT_PARAM Bit Assignment

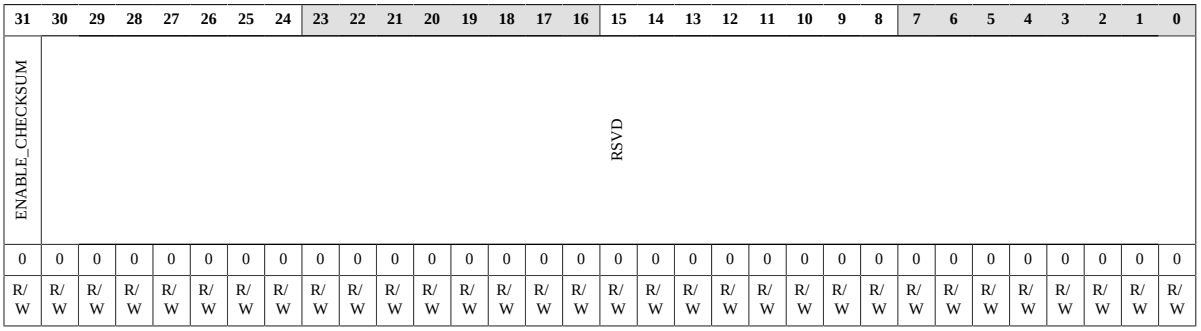


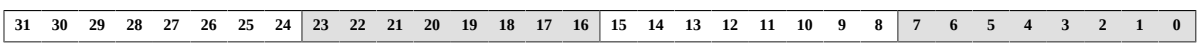
Table 1.501. CMD_ENC_REPORT_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31]	ENABLE_CHECKSUM	R/W	It eanbles to report check sum data of source frame buffer and reference frame buffer for debugging purpose.	0x0
[30:0]	RSVD	R/W	Reserved	0x0

1.2.4.2.10.6. CMD_ENC_REPORT_ENDIAN (0x0000012C)

Report buffer endianness

Table 1.502. CMD_ENC_REPORT_ENDIAN Bit Assignment



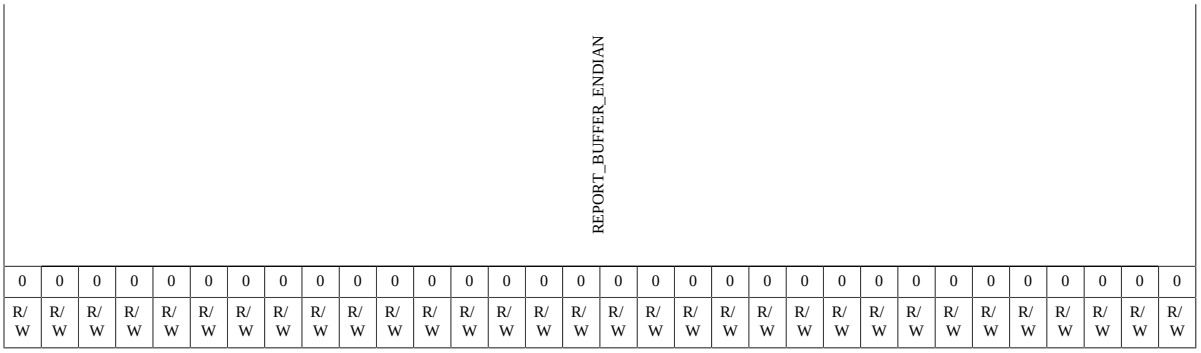


Table 1.503. CMD_ENC_REPORT_ENDIAN Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	REPORT_BUFFER_ENDIAN	R/W	Report buffer endianness	0x0

1.2.4.2.10.7. CMD_ENC_RESERVED (0x00000130)

Table 1.504. CMD_ENC_RESERVED Bit Assignment

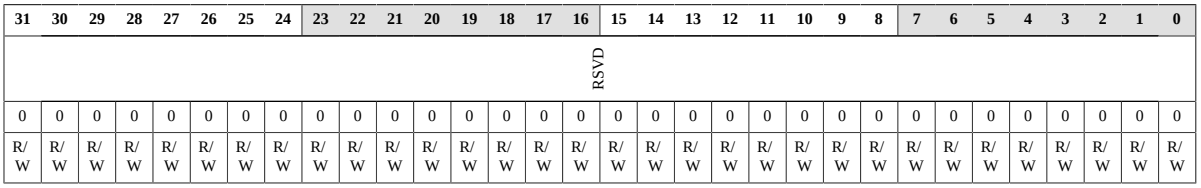


Table 1.505. CMD_ENC_RESERVED Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R/W	Reserved	0x0

1.2.4.2.10.8. CMD_ENC_CUSTOM_MAP_OPTION_PARAM (0x00000138)

Custom map parameters

Table 1.506. CMD_ENC_CUSTOM_MAP_OPTION_PARAM Bit Assignment

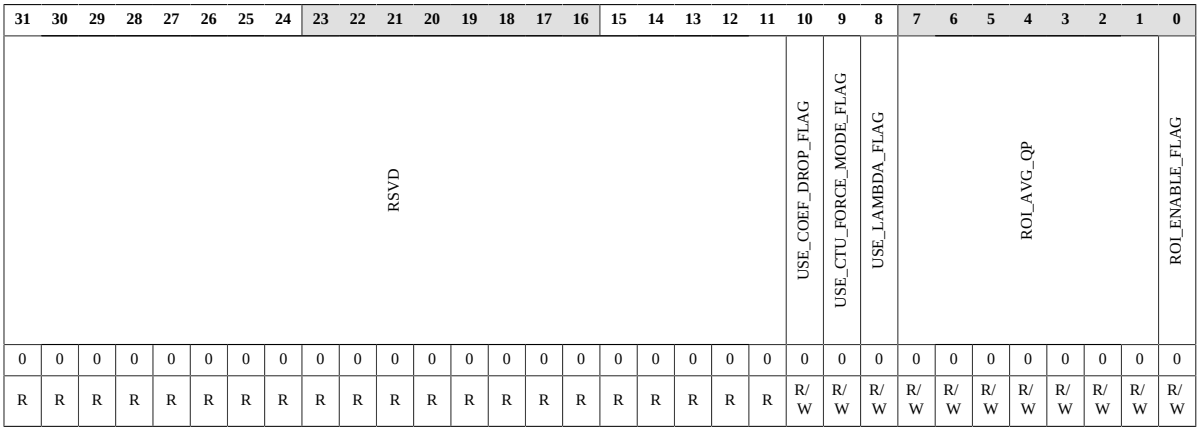


Table 1.507. CMD_ENC_CUSTOM_MAP_OPTION_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:11]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[10]	USE_COEF_DROP_FLAG	R/W	It uses a coefficient drop flag in CTU unit	0x0
[9]	USE_CTU_FORCE_MODE_FLAG	R/W	It forces intra, force skip in CTU unit.	0x0
[8]	USE_LAMBDA_FLAG	R/W	It uses sub-CTU lambda map.	0x0
[7:1]	ROI_AVG_QP	R/W	The average QP value for ROI	0x0
[0]	ROI_ENABLE_FLAG	R/W	It uses ROI map which holds a QP value for each sub-CTU. This is given by host .	0x0

1.2.4.2.10.9. CMD_ENC_CUSTOM_MAP_OPTION_ADDR (0x0000013C)

Custom map address

Table 1.508. CMD_ENC_CUSTOM_MAP_OPTION_ADDR Bit Assignment

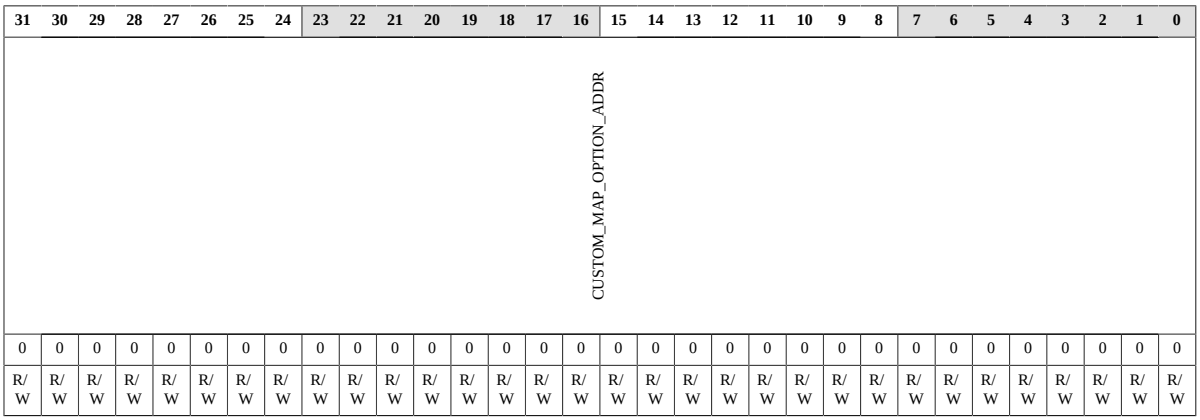


Table 1.509. CMD_ENC_CUSTOM_MAP_OPTION_ADDR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CUSTOM_MAP_OPTION_ADDR	R/W	Address of custom map	0x0

1.2.4.2.10.10. CMD_ENC_SRC_PIC_IDX (0x00000144)

GOP structure designation (MANDATORY)

Table 1.510. CMD_ENC_SRC_PIC_IDX Bit Assignment

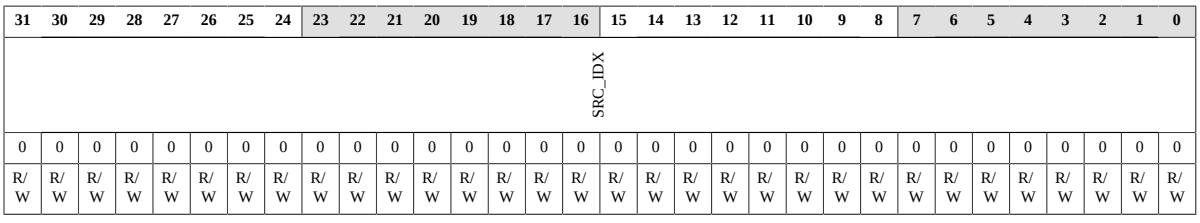


Table 1.511. CMD_ENC_SRC_PIC_IDX Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SRC_IDX	R/W	A buffer index of a source picture (-2 : no more source picture)	0x0

1.2.4.2.10.11. CMD_ENC_SRC_ADDR_Y (0x00000148)

Y component source address

Table 1.512. CMD_ENC_SRC_ADDR_Y Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_ADDR_Y																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.513. CMD_ENC_SRC_ADDR_Y Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SRC_ADDR_Y	R/W	An address of luma component of a source picture buffer An address of luma compressed source buffer when EN_CFRAME50 is 1	0x0

1.2.4.2.10.12. CMD_ENC_SRC_ADDR_U (0x0000014C)

Cb component source address

Table 1.514. CMD_ENC_SRC_ADDR_U Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_ADDR_U																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.515. CMD_ENC_SRC_ADDR_U Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SRC_ADDR_U	R/W	An address of Cb component of a source picture buffer An address of Cb compressed source buffer when EN_CFRAME50 is 1	0x0

1.2.4.2.10.13. CMD_ENC_SRC_ADDR_V (0x00000150)

Cr component source address

Table 1.516. CMD_ENC_SRC_ADDR_V Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_ADDR_V																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.517. CMD_ENC_SRC_ADDR_V Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SRC_ADDR_V	R/W	An address of Cr component of a source picture buffer An address of Cr compressed source buffer when EN_CFRAME50 is 1	0x0

1.2.4.2.10.14. CMD_ENC_SRC_STRIDE (0x00000154)

Stride of source picture

Table 1.518. CMD_ENC_SRC_STRIDE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_Y_STRIDE																SRC_C_STRIDE															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.519. CMD_ENC_SRC_STRIDE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	SRC_Y_STRIDE	R/W	A luma stride of source picture in pixel	0x0
[15:0]	SRC_C_STRIDE	R/W	A chroma stride of source picture in pixel	0x0

1.2.4.2.10.15. CMD_ENC_SRC_FORMAT (0x00000158)

Format of source picture

Table 1.520. CMD_ENC_SRC_FORMAT Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																						SRC_ENDIAN				SRC_PIXEL_FORMAT			SRC_FRAME_FORMAT		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.521. CMD_ENC_SRC_FORMAT Field Description

Bit	Name	Type	Function	Reset Value
[31:10]	RSVD	R	Reserved	0x0
[9:6]	SRC_ENDIAN	R/W	Endianness of source picture	0x0
[5:3]	SRC_PIXEL_FORMAT	R/W	[2] left justified [1:0]	0x0

Bit	Name	Type	Function	Reset Value
			0 : 8bit 1 : 16bit (1 pixel / 2 byte) 2 : 32bit (3 pixel / 4 byte)	
[2:0]	SRC_FRAME_FORMAT	R/W	0 : Planar 1 : Tiled Sub-CTU frame map - within a sub-CTU(32x32), the first row of 16x32 is read in vertical direction and then the second row of 16x32 is read. 2 : CbCr interleaved (NV12) 3 : CrCb interleaved (NV21) 4 : Packed mode (YUYV) 5 : Packed mode (YVYU) 6 : Packed mode (UYVY) 7 : Packed mode (VYUY)	0x0

1.2.4.2.10.16. CMD_ENC_SRC_AXI_SEL (0x00000160)

Selection of AXI port to load source pictures

Table 1.522. CMD_ENC_SRC_AXI_SEL Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																															PRP_GDI_SEL
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW

Table 1.523. CMD_ENC_SRC_AXI_SEL Field Description

Bit	Name	Type	Function	Reset Value
[31:1]	RSVD	R	Reserved	0x0
[0]	PRP_GDI_SEL	RW	0 : PRP port (default) 1 : Primary port	0x0

1.2.4.2.10.17. CMD_ENC_CODE_OPTION (0x00000164)

This register sets NAL coding options. By using this register, user can make (additional) NALs for the frame. In case of SVAC ENC, VPU only support EOS, SPS, VCL and IMPLICITY_HEADER_ENCODE flag only

Table 1.524. CMD_ENC_CODE_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																								FILLER	EOB	EOS	AUD	PPS	SPS	VPS	VCL	IMPLICITLY_HEADER_ENCODE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.525. CMD_ENC_CODE_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:9]	RSVD	R/W	Reserved	0x0
[8]	FILLER	R/W	A flag to encode FILLER NAL unit explicitly (the size is calculated with RC process)	0x0
[7]	EOB	R/W	A flag to encode EOB nal unit explicitly	0x0
[6]	EOS	R/W	A flag to encode EOS nal unit explicitly	0x0
[5]	AUD	R/W	A flag to encode AUD nal unit explicitly	0x0
[4]	PPS	R/W	A flag to encode PPS nal unit explicitly	0x0
[3]	SPS	R/W	A flag to encode SPS nal unit explicitly	0x0
[2]	VPS	R/W	A flag to encode VPS nal unit explicitly	0x0
[1]	VCL	R/W	A flag to encode VCL nal unit explicitly	0x0
[0]	IMPLICITLY_HEADER_ENCODE	R/W	A flag to enable to encode a header or headers (VPS, SPS, PPS) implicitly for generating bit-streams conforming to specification	0x0

1.2.4.2.10.18. CMD_ENC_PIC_PARAM (0x00000168)

Picture level encoding parameters

Table 1.526. CMD_ENC_PIC_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					DISABLE_DBK_IDC		EN_FORCE_COEF_DROP	FORCE_PIC_TYPE			USE_FORCE_PIC_TYPE	FORCE_PIC_QP_B				FORCE_PIC_QP_P								FORCE_PIC_QP_I						USE_FORCE_PIC_QP	PIC_SKIP_FLAG
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.527. CMD_ENC_PIC_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:27]	RSVD	R/W	Reserved	0x0
[26:25]	DISABLE_DBK_IDC	R/W	It disables the in-loop deblocking filter idc. NOTE: This is only valid in H.264/AVC decoder. In SVC mode (HEVC or SVAC), [25] can be used as SVC_LAYER_FLAG 0 : base layer picture (default) 1 : enhance layer picture	0x0
[24]	EN_FORCE_COEF_DROP	R/W	It forces all coefficients to be zero after TQ (to be dropped).	0x0
[23:21]	FORCE_PIC_TYPE	R/W	A force picture type (I, P, B, IDR, CRA) or an irap type 0 : I picture	0x0

Bit	Name	Type	Function	Reset Value
			1 : P picture 2 : B picture 3 : IDR picture 4 : CRA picture NOTE: In case of SVAC_ENC, VPU supports I, P, B picture only	
[20]	USE_FORCE_PIC_TYPE	R/W	A flag whether to use a force picture type or an irap type	0x0
[19:14]	FORCE_PIC_QP_B	R/W	A force picture quantization parameter for B picture (0 ~ 51) NOTE: In case of SVAC_ENC, Qp range can be 0 ~ 63	0x0
[13:8]	FORCE_PIC_QP_P	R/W	A force picture quantization parameter for P picture (0 ~ 51) NOTE: In case of SVAC_ENC, Qp range can be 0 ~ 63	0x0
[7:2]	FORCE_PIC_QP_I	R/W	A force picture quantization parameter for I picture (0 ~ 51) NOTE: In case of SVAC_ENC, Qp range can be 0 ~ 63	0x0
[1]	USE_FORCE_PIC_QP	R/W	A flag to use a force picture quantization parameter	0x0
[0]	PIC_SKIP_FLAG	R/W	A flag to skip the current picture	0x0

1.2.4.2.10.19. CMD_ENC_LONGTERM_PIC (0x0000016C)

Longterm picture setting

Table 1.528. CMD_ENC_LONGTERM_PIC Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																														USE_REF_LONGTERM_PIC	USE_SRC_LONGTERM_PIC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W

Table 1.529. CMD_ENC_LONGTERM_PIC Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1]	USE_REF_LONGTERM_PIC	R/W	A flag to use a longterm reference picture in DPB when encoding the current picture	0x0
[0]	USE_SRC_LONGTERM_PIC	R/W	A flag for the current picture to be used as a longterm reference picture later when other picture's encoding	0x0

1.2.4.2.10.20. CMD_ENC_WP_PIXEL_SIGMA_Y (0x00000170)

Luma pixel sigma for weighted prediction

Table 1.530. CMD_ENC_WP_PIXEL_SIGMA_Y Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

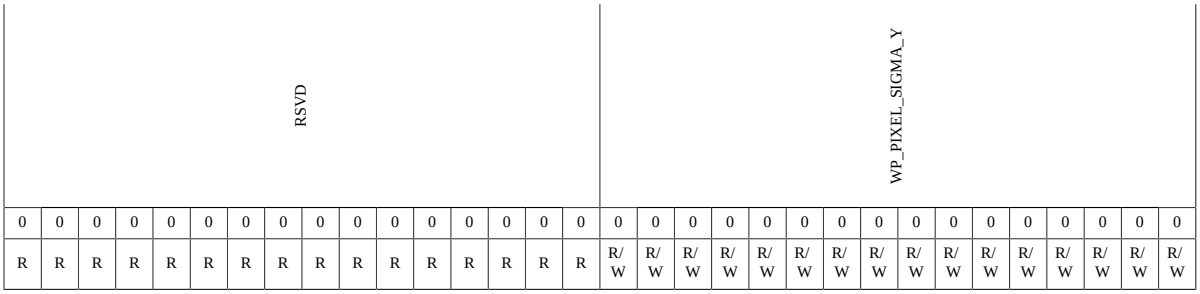


Table 1.531. CMD_ENC_WP_PIXEL_SIGMA_Y Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:0]	WP_PIXEL_SIGMA_Y	R/W	Pixel standard deviation of luma component for weighted prediction	0x0

1.2.4.2.10.21. CMD_ENC_WP_PIXEL_SIGMA_C (0x00000174)

Chroma pixel sigma for weighted prediction

Table 1.532. CMD_ENC_WP_PIXEL_SIGMA_C Bit Assignment

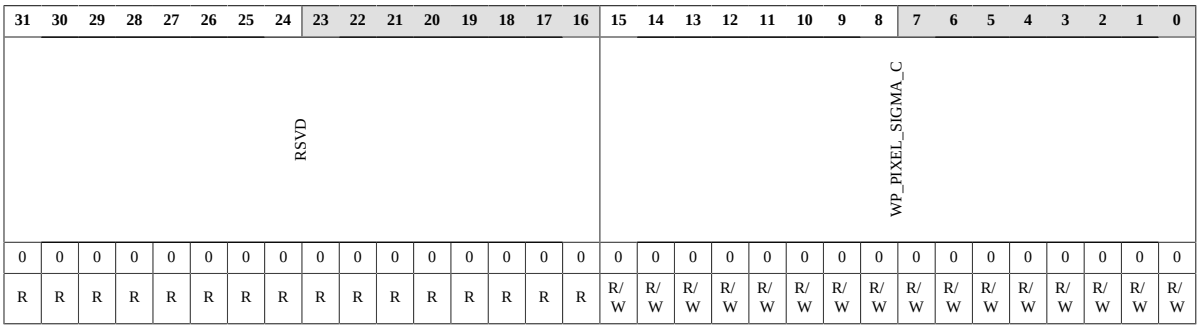


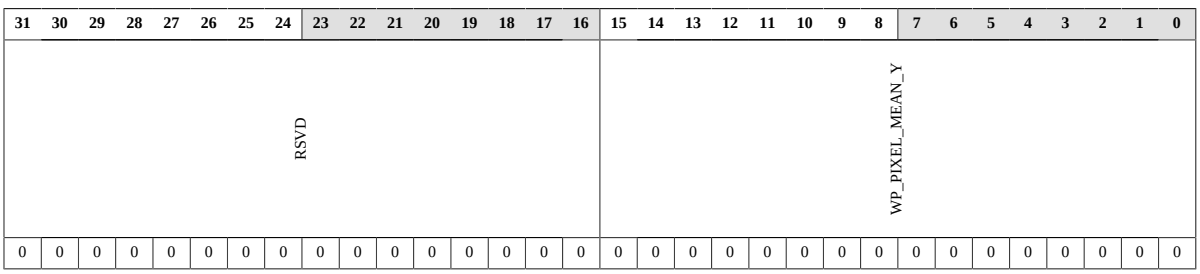
Table 1.533. CMD_ENC_WP_PIXEL_SIGMA_C Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:0]	WP_PIXEL_SIGMA_C	R/W	Pixel standard deviation of chroma component for weighted prediction	0x0

1.2.4.2.10.22. CMD_ENC_WP_PIXEL_MEAN_Y (0x00000178)

Luma pixel mean value for weighted prediction

Table 1.534. CMD_ENC_WP_PIXEL_MEAN_Y Bit Assignment



R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.535. CMD_ENC_WP_PIXEL_MEAN_Y Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:0]	WP_PIXEL_MEAN_Y	R/W	Pixel mean value of luma component for weighted prediction	0x0

1.2.4.2.10.23. CMD_ENC_WP_PIXEL_MEAN_C (0x0000017C)

Chroma pixel mean value for weighted prediction

Table 1.536. CMD_ENC_WP_PIXEL_MEAN_C Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																WP_PIXEL_MEAN_C															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.537. CMD_ENC_WP_PIXEL_MEAN_C Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:0]	WP_PIXEL_MEAN_C	R/W	Pixel mean value of chroma component for weighted prediction	0x0

1.2.4.2.10.24. CMD_ENC_PIC_LF_PARAM_0 (0x00000180)

For SVAC Encoding only

Table 1.538. CMD_ENC_PIC_LF_PARAM_0 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	LF_SHARPNESS_LEVEL			LF_MODE_DELTA							LF_REF_DELTA_L1							LF_REF_DELTA_L0							LF_DELTA_INTRA							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.539. CMD_ENC_PIC_LF_PARAM_0 Field Description

Bit	Name	Type	Function	Reset Value
[31]	RSVD	R/W	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[30:28]	LF_SHARPNESS_LEVEL	R/W	Specifies the sharpness level of filter. (0 ~ 7)	0x0
[27:21]	LF_MODE_DELTA	R/W	Specifies a delta value of filter level according to intra/inter mode.	0x0
[20:14]	LF_REF_DELTA_L1	R/W	Specifies a delta value of filter level for Ref1 frame. (Optional Ref)	0x0
[13:7]	LF_REF_DELTA_L0	R/W	Specifies a delta value of filter level for Ref0 frame. (Dynamic Ref)	0x0
[6:0]	LF_DELTA_INTRA	R/W	Specifies a delta value of filter level for key frame.	0x0

1.2.4.2.10.25. CMD_ENC_PIC_LF_PARAM_1 (0x00000184)

For SVAC Encoding only

Table 1.540. CMD_ENC_PIC_LF_PARAM_1 Bit Assignment

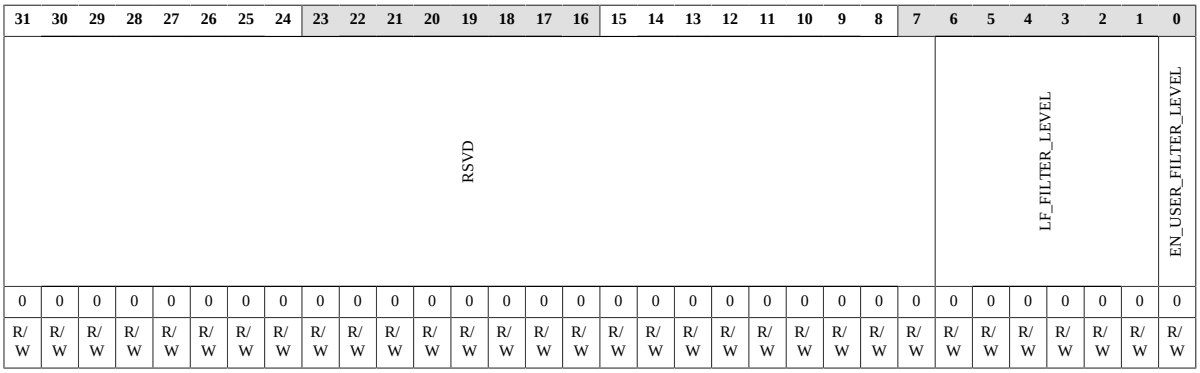


Table 1.541. CMD_ENC_PIC_LF_PARAM_1 Field Description

Bit	Name	Type	Function	Reset Value
[31:7]	RSVD	R/W	Reserved	0x0
[6:1]	LF_FILTER_LEVEL	R/W	Specifies the loop filter level. (0 ~ 63)	0x0
[0]	EN_USER_FILTER_LEVEL	R/W	Enables to set the user filter level.	0x0

1.2.4.2.10.26. CMD_ENC_SRC_CF50_ADDR_Y_OFFSET (0x00000188)

Base address of offset table Y for CF50 when source image format is CF50

Note This is valid only when CF50 hardware is included in the released IP.

Table 1.542. CMD_ENC_SRC_CF50_ADDR_Y_OFFSET Bit Assignment

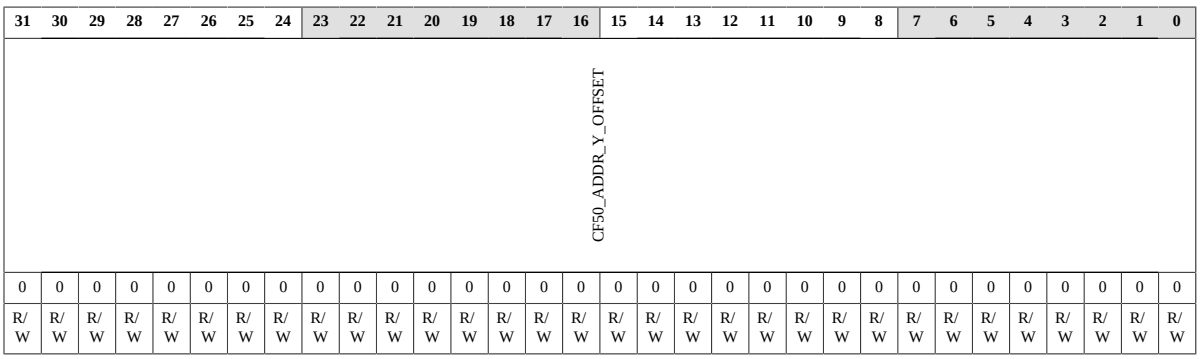


Table 1.543. CMD_ENC_SRC_CF50_ADDR_Y_OFFSET Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CF50_ADDR_Y_OFFSET	R/W	Base address of Offset table Y NOTE: Address should be aligned to word size.	0x0

1.2.4.2.10.27. CMD_ENC_SRC_CF50_ADDR_CB_OFFSET (0x0000018C)

Base address of offset table CB for CF50 when source image format is CF50

Note This is valid only when CF50 hardware is included in the released IP.

Table 1.544. CMD_ENC_SRC_CF50_ADDR_CB_OFFSET Bit Assignment

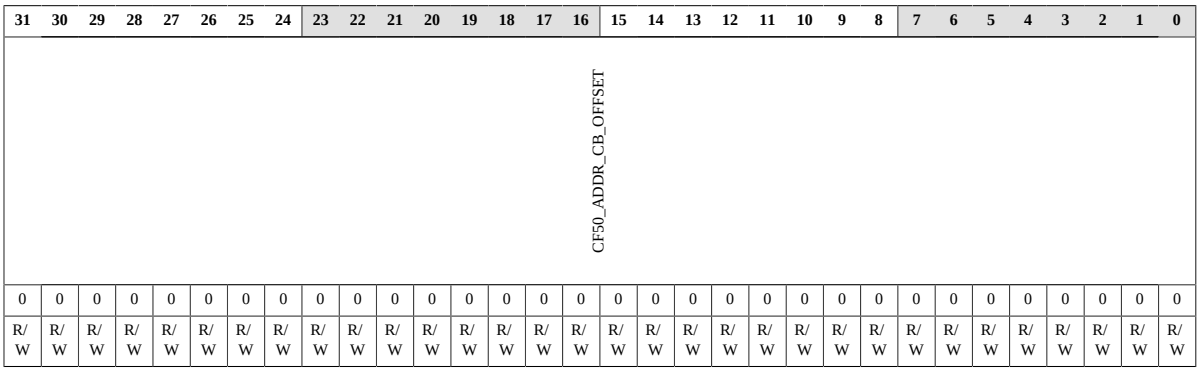


Table 1.545. CMD_ENC_SRC_CF50_ADDR_CB_OFFSET Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CF50_ADDR_CB_OFFSET	R/W	Base address of Offset table CB NOTE: Address should be aligned to word size.	0x0

1.2.4.2.10.28. CMD_ENC_SRC_CF50_ADDR_CR_OFFSET (0x00000190)

Base address of offset table CR for CF50 when source image format is CF50

Note This is valid only when CF50 hardware is included in the released IP.

Table 1.546. CMD_ENC_SRC_CF50_ADDR_CR_OFFSET Bit Assignment

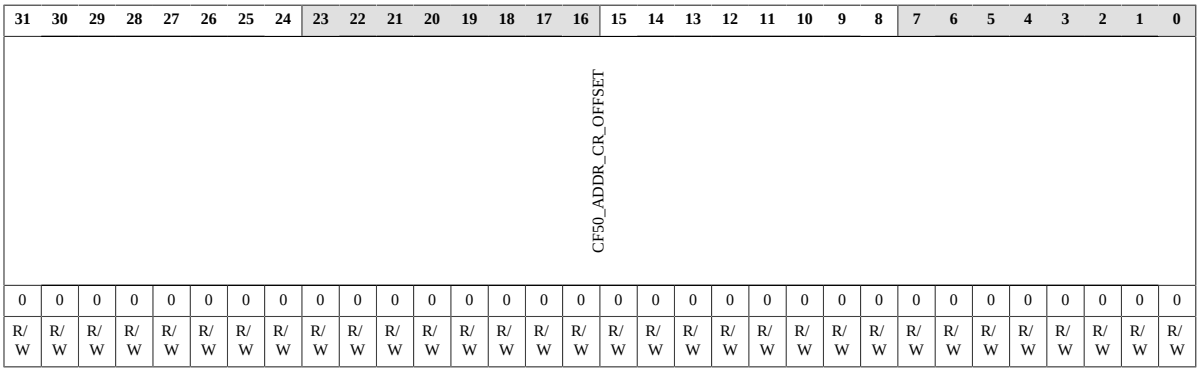


Table 1.547. CMD_ENC_SRC_CF50_ADDR_CR_OFFSET Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CF50_ADDR_CR_OFFSET	R/W	Base address of Offset table CR NOTE: Address should be aligned to word size.	0x0

1.2.4.2.11. QUERY(GET_VPU_INFO) Command Parameter Registers

These are command I/O registers where Host processor can set arguments for the QUERY command with 1 of CMD_QUERY_OPTION or get return values.

1.2.4.2.11.1. CMD_QUERY_OPTION (0x00000104)

QUERY command option

Table 1.548. CMD_QUERY_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										QUERY_OPTION					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.549. CMD_QUERY_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:6]	RSVD	R	Reserved	0x0
[5:0]	QUERY_OPTION	R/W	**0x00: GET_VPU_INFO** 0x02: GET_RESULT 0x03: UPDATE_DISP_IDC 0x04: GET_BW_RESULT 0x05: GET_BS_RD_PTR 0x06: GET_BS_WR_PTR	0x0

1.2.4.2.11.2. RET_QUERY_FW_VERSION (0x00000118)

Table 1.550. RET_QUERY_FW_VERSION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET_FW_VERSION																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.551. RET_QUERY_FW_VERSION Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RET_FW_VERSION	R	Firmware version (internal use only)	0x0

1.2.4.2.11.3. RET_QUERY_PRODUCT_NAME (0x0000011C)

Table 1.552. RET_QUERY_PRODUCT_NAME Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

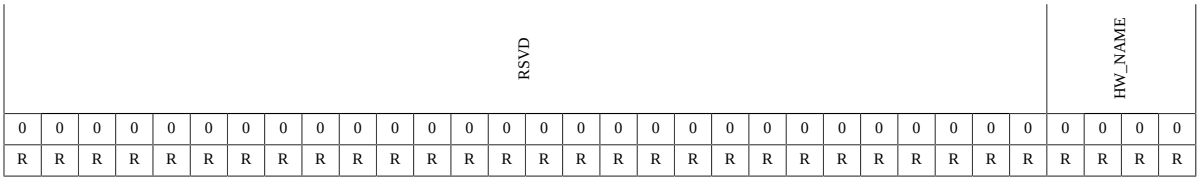


Table 1.553. RET_QUERY_PRODUCT_NAME Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	HW_NAME	R	VPU hardware product name It always returns "WAVE" for WAVE5 series IP product.	0x0

1.2.4.2.11.4. RET_QUERY_PRODUCT_VERSION (0x00000120)

Table 1.554. RET_QUERY_PRODUCT_VERSION Bit Assignment

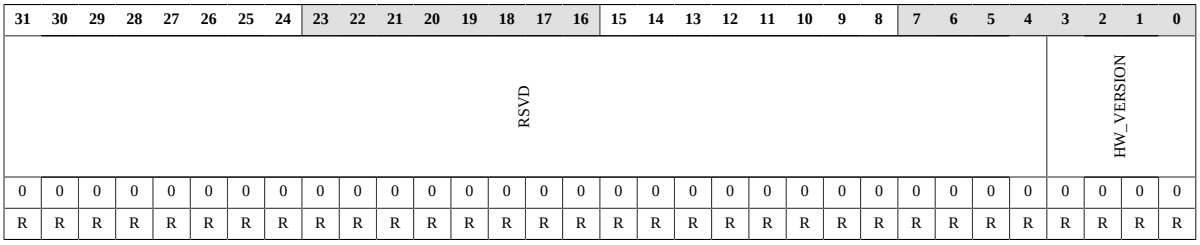


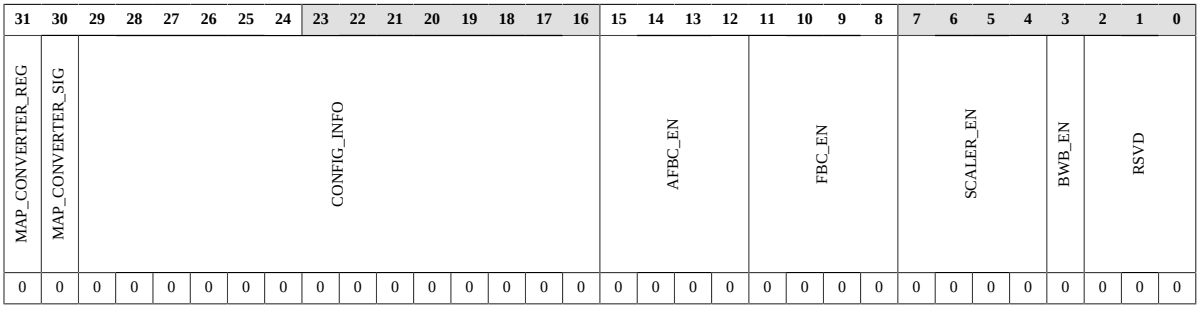
Table 1.555. RET_QUERY_PRODUCT_VERSION Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	HW_VERSION	R	VPU hardware product version It returns as follows: 0x5120 for WAVE512 0x5150 for WAVE515 0x5110 for WAVE511 0x5200 for WAVE520 0x5250 for WAVE525 0x5210 for WAVE521	0x0

1.2.4.2.11.5. RET_QUERY_STD_DEF0 (0x00000124)

System configuration information (internal use only) such as external memory interface, external model information, and output support

Table 1.556. RET_QUERY_STD_DEF0 Bit Assignment



R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.557. RET_QUERY_STD_DEF0 Field Description

Bit	Name	Type	Function	Reset Value
[31]	MAP_CONVERTER_REG	R	Extenral Memory interface Definition Use of register-based map converter	0x0
[30]	MAP_CONVERTER_SIG	R	Extenral Memory interface Definition Use of signal-based map converter	0x0
[29:16]	CONFIG_INFO	R	[21] std_switch_en [20] bg_detect [19] 3dnr_en [18] one_axi_en [17] SEC_AXI_EN [16] bus_info. GDI	0x0
[15:12]	AFBC_EN	R	Output Support {AFBC enable, AFBC version ID [2:0]}	0x0
[11:8]	FBC_EN	R	Output Support {FBC enable, FBC version ID[2:0]}	0x0
[7:4]	SCALER_EN	R	Output Support {SCALER enable, SCALER version ID[2:0]}	0x0
[3]	BWB_EN	R	BWB enable flag	0x0
[2:0]	RSVD	R	Reserved	0x0

1.2.4.2.11.6. RET_QUERY_STD_DEF1 (0x00000128)

General hardware configuration information (internal use only)

Table 1.558. RET_QUERY_STD_DEF1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERNAL_BLOCK																FEATURE_SET															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.559. RET_QUERY_STD_DEF1 Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	INTERNAL_BLOCK	R	Internal block [31:24] Enabled Support Logics (CU_REPORT, GCU etc.) [26] MULTICORE_EN [25] GCU_EN [24] CU_REPORT [23:16] MULTICORE_PRESENT_INFO	0x0
[15:0]	FEATURE_SET	R	Feature set (based on revision, FW writes) [15] Support bandwidth optimization [14:2] reserved [1] std_vp9_en [0] std_hevc_en	0x0

1.2.4.2.11.7. RET_QUERY_CONF_FEATURE (0x0000012C)

Table 1.560. RET_QUERY_CONF_FEATURE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONFIG_FEATURE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.561. RET_QUERY_CONF_FEATURE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CONFIG_FEATURE	R	Each flag shows supported codec standard in the WAVE5 series (internal use only) [5] VP9_DEC_PROFILE2 [4] VP9_DEC_PROFILE0 [3] Reserved for HEVC_ENC_MAIN10 [2] Reserved for HEVC_ENC_MAIN [1] HEVC_DEC_MAIN10 [0] HEVC_DEC_MAIN	0x0

1.2.4.2.11.8. RET_QUERY_CONF_DATE (0x00000130)

Table 1.562. RET_QUERY_CONF_DATE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW_DATE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.563. RET_QUERY_CONF_DATE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	HW_DATE	R	Configuation information 0 (internal use only) This register has the configuration date for the package.	0x0

1.2.4.2.11.9. RET_QUERY_CONF_REVISION (0x00000134)

Table 1.564. RET_QUERY_CONF_REVISION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW_REVISION																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.565. RET_QUERY_CONF_REVISION Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	HW_REVISION	R	Configuation information 1 (internal use only) This register has revision information for the package.	0x0

1.2.4.2.11.10. RET_QUERY_CONF_TYPE (0x00000138)

Table 1.566. RET_QUERY_CONF_TYPE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW_TYPE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.567. RET_QUERY_CONF_TYPE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	HW_TYPE	R	Configuation information 2 This register has configuration type for the package.	0x0

1.2.4.2.11.11. RET_QUERY_PRODUCT_ID (0x0000013C)

Table 1.568. RET_QUERY_PRODUCT_ID Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_ID																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.569. RET_QUERY_PRODUCT_ID Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PRODUCT_ID	R	The product id (internal use only)	0x0

1.2.4.2.11.12. RET_QUERY_CUSTOMER_ID (0x00000140)

Table 1.570. RET_QUERY_CUSTOMER_ID Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUSTOMER_ID																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.571. RET_QUERY_CUSTOMER_ID Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CUSTOMER_ID	R	The customer id (internal use only)	0x0

1.2.4.2.12. QUERY(GET_RESULT) Command Parameter Registers for Encoder

These are command I/O registers where Host processor can set arguments for the QUERY command with 2 of CMD_QUERY_OPTION or get return values.

1.2.4.2.12.1. CMD_QUERY_OPTION (0x00000104)

QUERY command option

Table 1.572. CMD_QUERY_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										QUERY_OPTION					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.573. CMD_QUERY_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:6]	RSVD	R	Reserved	0x0
[5:0]	QUERY_OPTION	R/W	0x00: GET_VPU_INFO **0x02: GET_RESULT** 0x03: UPDATE_DISP_IDC 0x04: GET_BW_RESULT 0x05: GET_BS_RD_PTR 0x06: GET_BS_WR_PTR	0x0

1.2.4.2.12.2. RET_QUERY_ENC_RD_PTR (0x00000114)

Bistream buffer read pointer

Table 1.574. RET_QUERY_ENC_RD_PTR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD_PTR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.575. RET_QUERY_ENC_RD_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RD_PTR	R	The start address of CPB where bistream writes	0x0

1.2.4.2.12.3. RET_QUERY_ENC_WR_PTR (0x00000118)

Bistream buffer write pointer

Table 1.576. RET_QUERY_ENC_WR_PTR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

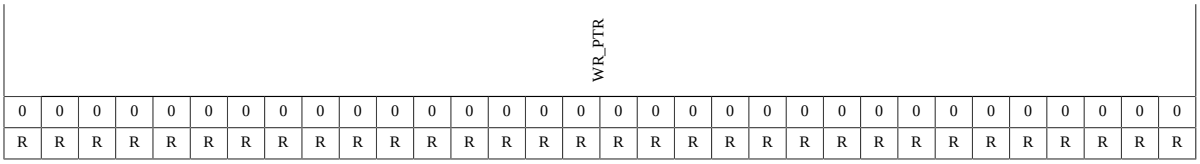


Table 1.577. RET_QUERY_ENC_WR_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	WR_PTR	R	The end address of CPB where bistream writes	0x0

1.2.4.2.12.4. RET_QUERY_ENC_NUM_REQUIRED_FB (0x0000011C)

Table 1.578. RET_QUERY_ENC_NUM_REQUIRED_FB Bit Assignment

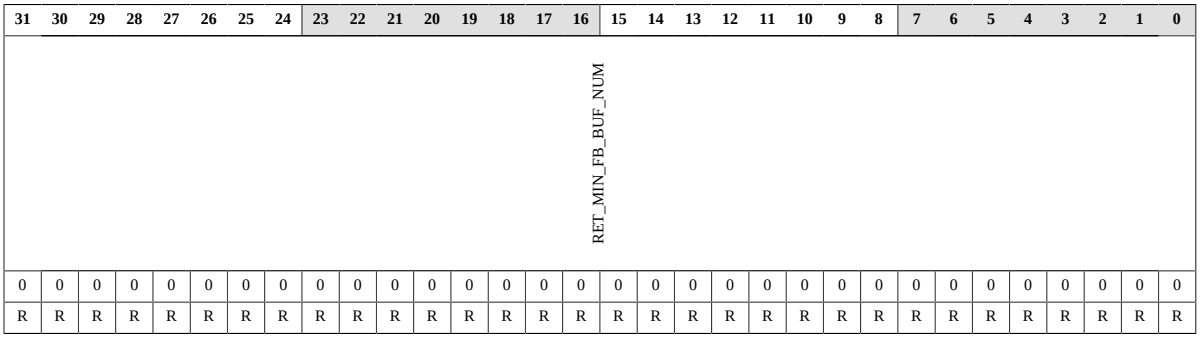


Table 1.579. RET_QUERY_ENC_NUM_REQUIRED_FB Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RET_MIN_FB_BUF_NUM	R	The minimum number of reference frame buffer for encoding (1 ~ 16)	0x0

1.2.4.2.12.5. RET_QUERY_MIN_SRC_BUF_NUM (0x00000120)

Table 1.580. RET_QUERY_MIN_SRC_BUF_NUM Bit Assignment

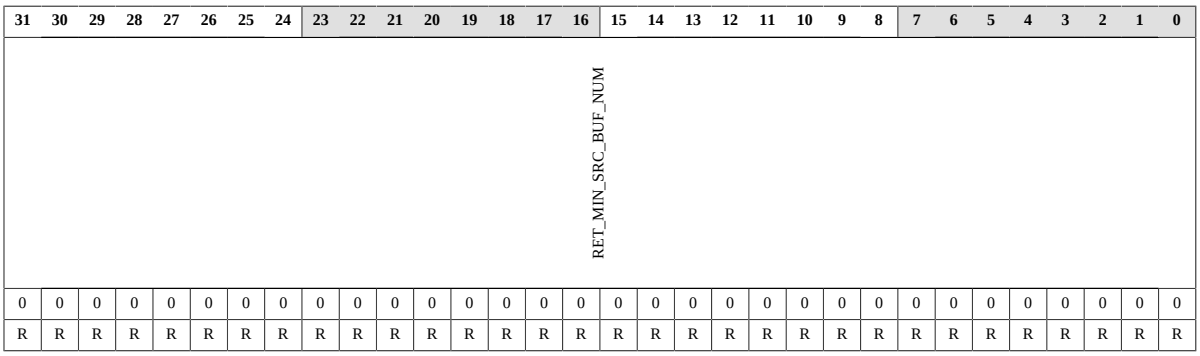


Table 1.581. RET_QUERY_MIN_SRC_BUF_NUM Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RET_MIN_SRC_BUF_NUM	R	The minimum number of source frame buffer for encoding (1 ~ 16)	0x0

1.2.4.2.12.6. RET_QUERY_ENC_PIC_TYPE (0x00000124)

Encoded picture type

Table 1.582. RET_QUERY_ENC_PIC_TYPE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																PIC_TYPE															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.583. RET_QUERY_ENC_PIC_TYPE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:0]	PIC_TYPE	R	The encoded picture type 0 : I slice 1 : P slice 2 : B slice	0x0

1.2.4.2.12.7. RET_QUERY_ENC_PIC_POC (0x00000128)

Table 1.584. RET_QUERY_ENC_PIC_POC Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIC_POC																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.585. RET_QUERY_ENC_PIC_POC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_POC	R	A POC value of the currently encoded picture	0x0

1.2.4.2.12.8. RET_QUERY_ENC_PIC_IDX (0x0000012C)

A frame buffer index of the encoded picture

Table 1.586. RET_QUERY_ENC_PIC_IDX Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIC_IDX																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.587. RET_QUERY_ENC_PIC_IDX Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_IDX	R	Greater than or equal to 0 : encoded picture buffer index -1 : encoding end -2 : encoding delay (default) -3 : header encoded only (no VCL) -4 : change of encoding parameter	0x0

1.2.4.2.12.9. RET_QUERY_ENC_PIC_SLICE_NUM (0x00000130)

Number of slice segments

Table 1.588. RET_QUERY_ENC_PIC_SLICE_NUM Bit Assignment

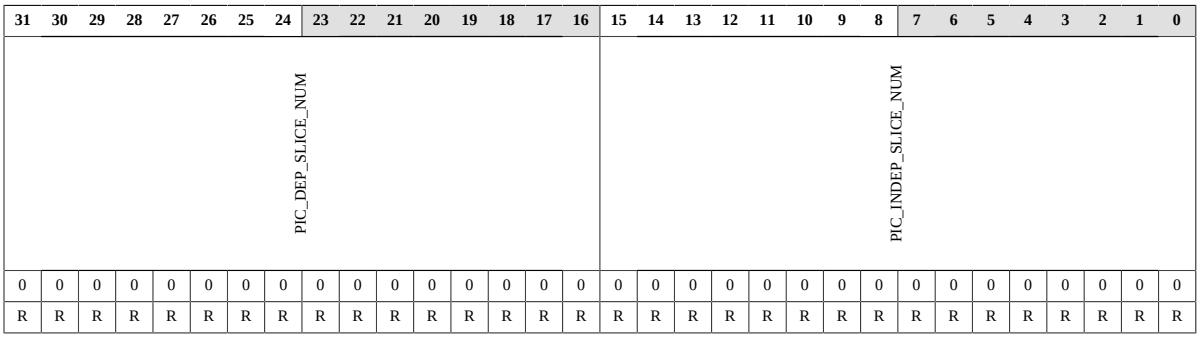


Table 1.589. RET_QUERY_ENC_PIC_SLICE_NUM Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	PIC_DEP_SLICE_NUM	R	The total dependent slice segment number in the encoded picture	0x0
[15:0]	PIC_INDEP_SLICE_NUM	R	The total independent slice segment number in the encoded picture	0x0

1.2.4.2.12.10. RET_QUERY_ENC_PIC_SKIP (0x00000134)

Table 1.590. RET_QUERY_ENC_PIC_SKIP Bit Assignment

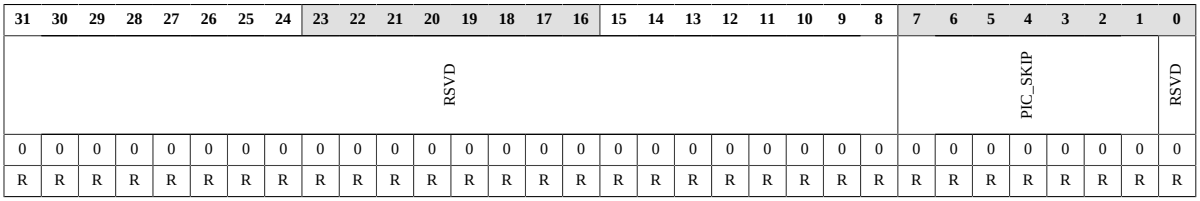


Table 1.591. RET_QUERY_ENC_PIC_SKIP Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:1]	PIC_SKIP	R	A picture skip flag	0x0
[0]	RSVD	R	Reserved	0x0

1.2.4.2.12.11. RET_QUERY_ENC_PIC_NUM_INTRA (0x00000138)

Table 1.592. RET_QUERY_ENC_PIC_NUM_INTRA Bit Assignment

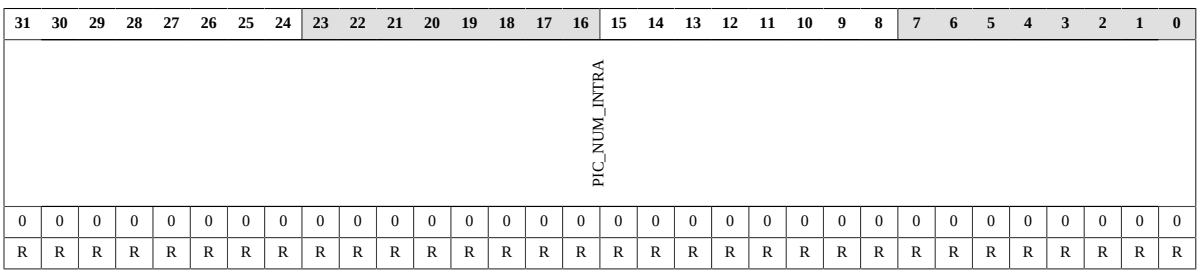


Table 1.593. RET_QUERY_ENC_PIC_NUM_INTRA Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_NUM_INTRA	R	The number of intra block in 8x8	0x0

1.2.4.2.12.12. RET_QUERY_ENC_PIC_NUM_MERGE (0x0000013C)

Table 1.594. RET_QUERY_ENC_PIC_NUM_MERGE Bit Assignment

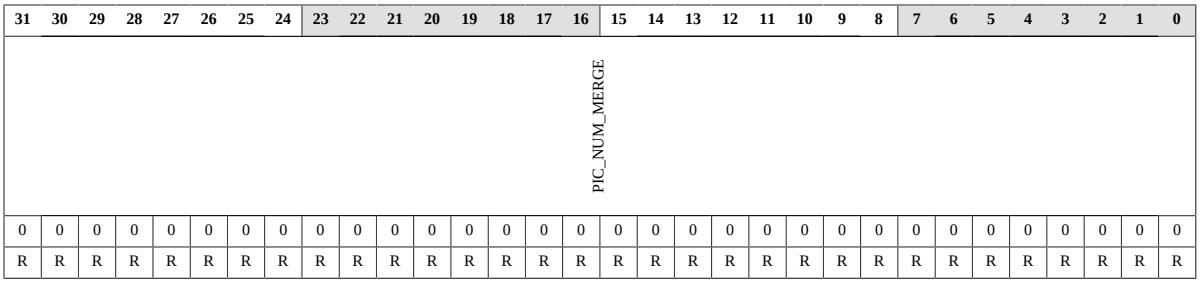


Table 1.595. RET_QUERY_ENC_PIC_NUM_MERGE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_NUM_MERGE	R	The number of merge block in 8x8	0x0

1.2.4.2.12.13. RET_QUERY_ENC_PIC_FLAG (0x00000140)

Table 1.596. RET_QUERY_ENC_PIC_FLAG Bit Assignment

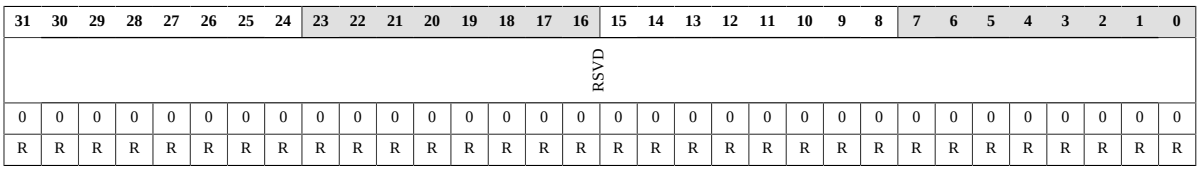


Table 1.597. RET_QUERY_ENC_PIC_FLAG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	R	Reserved	0x0

1.2.4.2.12.14. RET_QUERY_ENC_PIC_NUM_SKIP (0x00000144)

Table 1.598. RET_QUERY_ENC_PIC_NUM_SKIP Bit Assignment

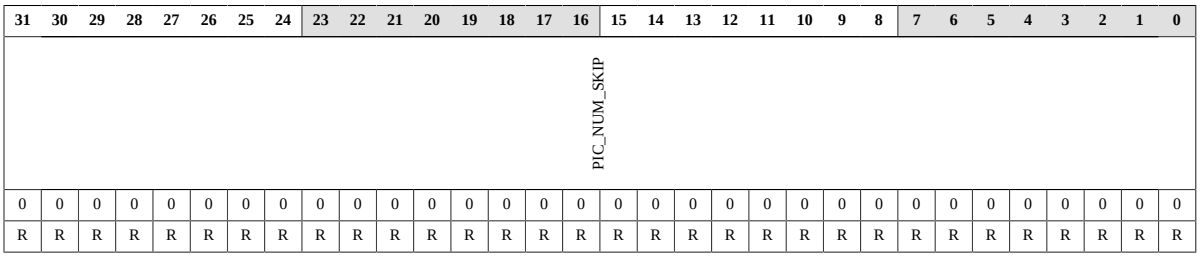


Table 1.599. RET_QUERY_ENC_PIC_NUM_SKIP Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_NUM_SKIP	R	The number of skip block in 8x8	0x0

1.2.4.2.12.15. RET_QUERY_ENC_PIC_AVG_CTU_QP (0x00000148)

Table 1.600. RET_QUERY_ENC_PIC_AVG_CTU_QP Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIC_AVG_CTU_QP																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.601. RET_QUERY_ENC_PIC_AVG_CTU_QP Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_AVG_CTU_QP	R	An average value of CTU QPs	0x0

1.2.4.2.12.16. RET_QUERY_ENC_PIC_BYTE (0x0000014C)

Table 1.602. RET_QUERY_ENC_PIC_BYTE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIC_BYTE																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.603. RET_QUERY_ENC_PIC_BYTE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_BYTE	R	The size of encoded picture in byte	0x0

1.2.4.2.12.17. RET_QUERY_ENC_GOP_PIC_IDX (0x00000150)

Table 1.604. RET_QUERY_ENC_GOP_PIC_IDX Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GOP_PIC_IDX																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.605. RET_QUERY_ENC_GOP_PIC_IDX Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	GOP_PIC_IDX	R	A picture index in GOP	0x0

1.2.4.2.12.18. RET_QUERY_ENC_USED_SRC_IDX (0x00000154)

Table 1.606. RET_QUERY_ENC_USED_SRC_IDX Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

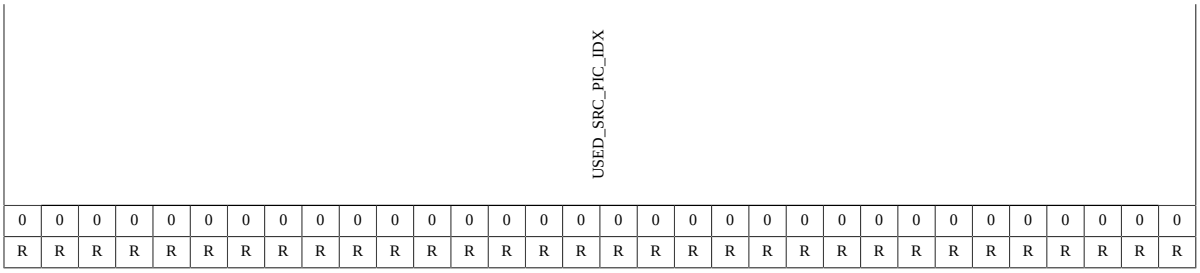


Table 1.607. RET_QUERY_ENC_USED_SRC_IDX Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	USED_SRC_PIC_IDX	R	A source buffer index of the encoded picture (-2 : encoding delay)	0x0

1.2.4.2.12.19. RET_QUERY_ENC_PIC_NUM (0x00000158)

Table 1.608. RET_QUERY_ENC_PIC_NUM Bit Assignment

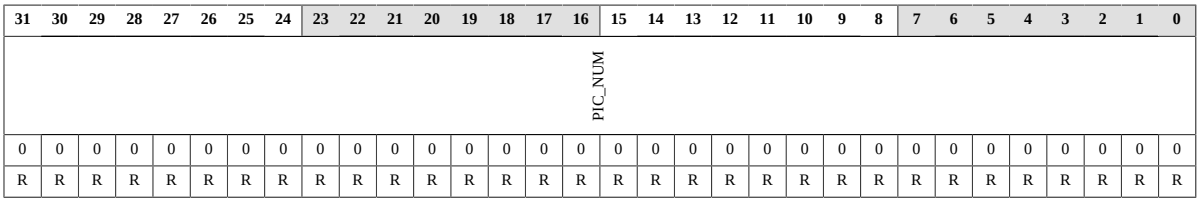


Table 1.609. RET_QUERY_ENC_PIC_NUM Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_NUM	R	The encoded picture number	0x0

1.2.4.2.12.20. RET_QUERY_ENC_NUT_0 (0x0000015C)

Encoded NAL unit type of VCL

Table 1.610. RET_QUERY_ENC_NUT_0 Bit Assignment

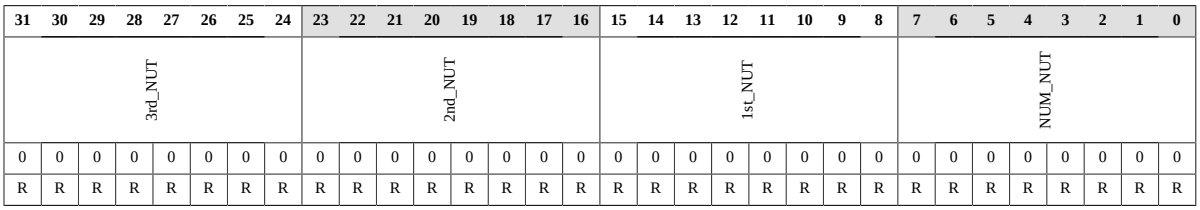


Table 1.611. RET_QUERY_ENC_NUT_0 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	3rd_NUT	R	The type of the 3rd NAL unit which has been encoded for the current command	0x0
[23:16]	2nd_NUT	R	The type of the 2nd NAL unit which has been encoded for the current command	0x0
[15:8]	1st_NUT	R	The type of the 1st NAL unit which has been encoded for the current command	0x0
[7:0]	NUM_NUT	R	The total number of encoded NAL unitfor the current command	0x0

1.2.4.2.12.21. RET_QUERY_ENC_NUT_1 (0x00000160)

Encoded NAL unit type of VCL

Table 1.612. RET_QUERY_ENC_NUT_1 Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7th_NUT								6th_NUT								5th_NUT								4th_NUT							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.613. RET_QUERY_ENC_NUT_1 Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	7th_NUT	R	The type of the 7th NAL unit which has been encoded for the current command	0x0
[23:16]	6th_NUT	R	The type of the 6th NAL unit which has been encoded for the current command	0x0
[15:8]	5th_NUT	R	The type of the 5th NAL unit which has been encoded for the current command	0x0
[7:0]	4th_NUT	R	The type of the 4th NAL unit which has been encoded for the current command	0x0

1.2.4.2.12.22. RET_QUERY_ENC_PIC_DIST_LOW (0x00000164)

Low 32bit SSD

Table 1.614. RET_QUERY_ENC_PIC_DIST_LOW Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIST_Y_LOW																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.615. RET_QUERY_ENC_PIC_DIST_LOW Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	DIST_Y_LOW	R	Low 32bit SSD between source Y picture and reconstructed Y picture	0x0

1.2.4.2.12.23. RET_QUERY_ENC_PIC_DIST_HIGH (0x00000168)

High 32bit SSD

Table 1.616. RET_QUERY_ENC_PIC_DIST_HIGH Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIST_Y_HIGH																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.617. RET_QUERY_ENC_PIC_DIST_HIGH Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	DIST_Y_HIGH	R	High 32bit SSD between source Y picture and reconstructed Y picture	0x0

1.2.4.2.12.24. RET_QUERY_ENC_MAX_LATENCY_PICTURES (0x0000016C)

The number of latency picture

Table 1.618. RET_QUERY_ENC_MAX_LATENCY_PICTURES Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAX_LATENCY_PIC																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.619. RET_QUERY_ENC_MAX_LATENCY_PICTURES Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	MAX_LATENCY_PIC	R	The number of picture delayed from reordering	0x0

1.2.4.2.12.25. RET_QUERY_ENC_SVC_LAYER (0x00000170)

Table 1.620. RET_QUERY_ENC_SVC_LAYER Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVC_LAYER																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.621. RET_QUERY_ENC_SVC_LAYER Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SVC_LAYER	R	SVC layer type of the encoded picture (SVAC encoder only) 0 : BL picture 1 : EL picture The default is 0 in non-SVC coding.	0x0

1.2.4.2.12.26. RET_QUERY_REPORT_PARAM (0x00000178)

Report parameters

Table 1.622. RET_QUERY_REPORT_PARAM Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

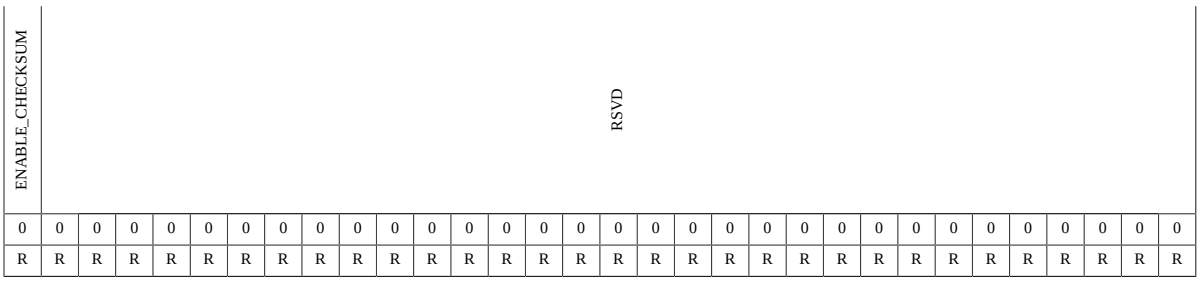


Table 1.623. RET_QUERY_REPORT_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31]	ENABLE_CHECKSUM	R	It eanbles to report check sum data of source frame buffer and reference frame buffer for debugging purpose.	0x0
[30:0]	RSVD	R	Reserved	0x0

1.2.4.2.12.27. RET_QUERY_ENC_REPORT_BASE (0x0000017C)

Report buffer base address

Table 1.624. RET_QUERY_ENC_REPORT_BASE Bit Assignment

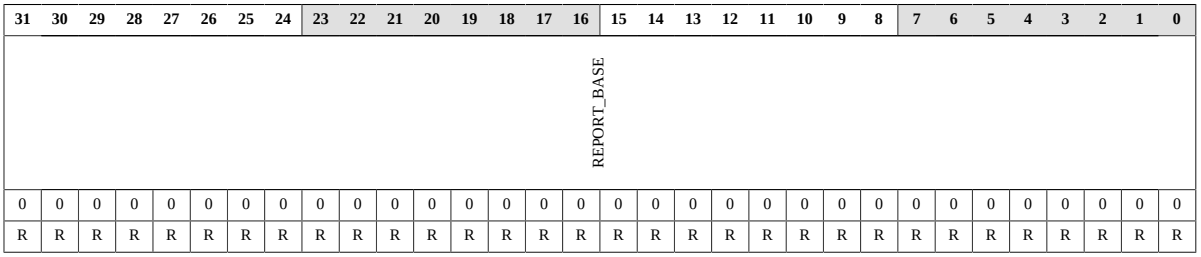


Table 1.625. RET_QUERY_ENC_REPORT_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	REPORT_BASE	R	Base address of CU report buffer CU report buffer holds CU header information.	0x0

1.2.4.2.12.28. RET_QUERY_HOST_CMD_TICK (0x000001B8)

Table 1.626. RET_QUERY_HOST_CMD_TICK Bit Assignment

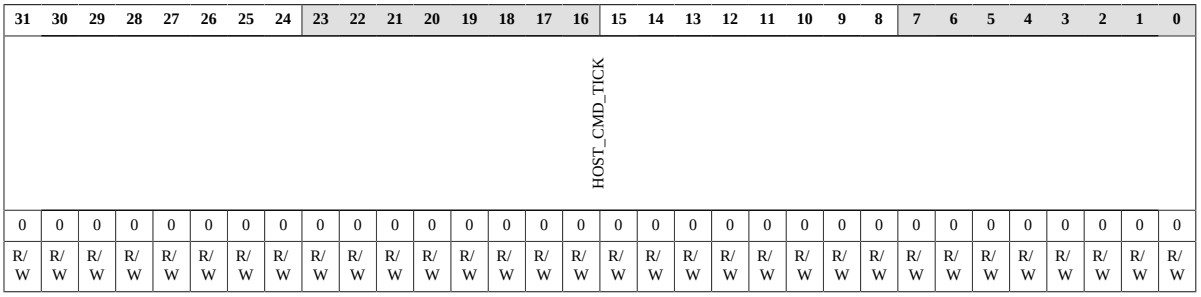


Table 1.627. RET_QUERY_HOST_CMD_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	HOST_CMD_TICK	R/W	Tick of ENC_PIC command for the picture	0x0

1.2.4.2.12.29. RET_QUERY_DEC_PREPARE_START_TICK (0x000001BC)

Table 1.628. RET_QUERY_DEC_PREPARE_START_TICK Bit Assignment

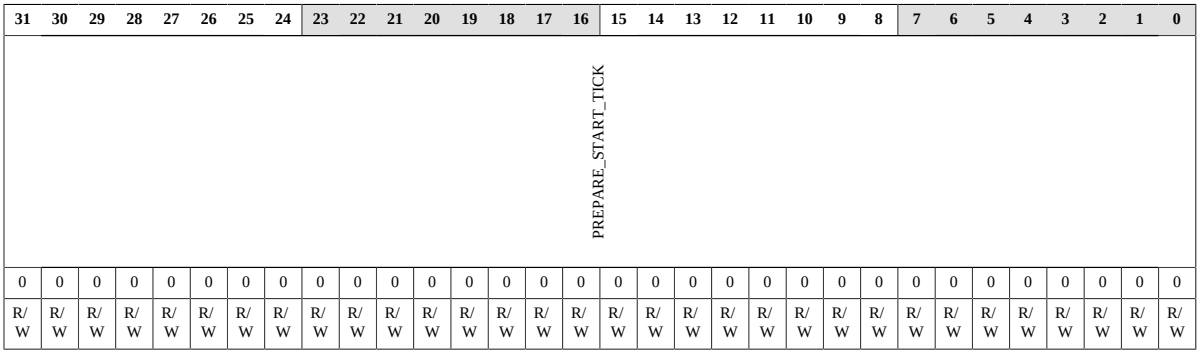


Table 1.629. RET_QUERY_DEC_PREPARE_START_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PREPARE_START_TICK	R/W	Start tick of preparing slices of the picture	0x0

1.2.4.2.12.30. RET_QUERY_DEC_PREPARE_END_TICK (0x000001C0)

Table 1.630. RET_QUERY_DEC_PREPARE_END_TICK Bit Assignment

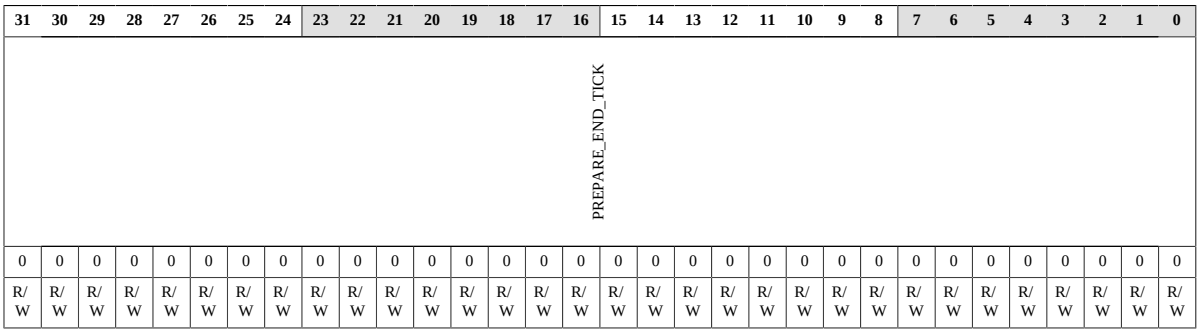
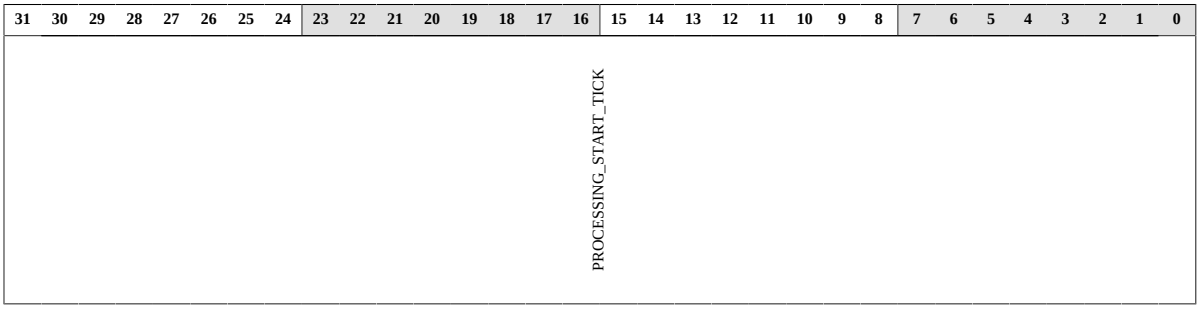


Table 1.631. RET_QUERY_DEC_PREPARE_END_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PREPARE_END_TICK	R/W	End tick of preparing slices of the picture	0x0

1.2.4.2.12.31. RET_QUERY_DEC_PROCESSING_START_TICK (0x000001C4)

Table 1.632. RET_QUERY_DEC_PROCESSING_START_TICK Bit Assignment



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.633. RET_QUERY_DEC_PROCESSING_START_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PROCESSING_START_TICK	R/W	Start tick of processing slices of the picture	0x0

1.2.4.2.12.32. RET_QUERY_DEC_PROCESSING_END_TICK (0x000001C8)

Table 1.634. RET_QUERY_DEC_PROCESSING_END_TICK Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PROCESSING_END_TICK																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.635. RET_QUERY_DEC_PROCESSING_END_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PROCESSING_END_TICK	R/W	End tick of processing slices of the picture	0x0

1.2.4.2.12.33. RET_QUERY_DEC_ENCODING_START_TICK (0x000001CC)

Table 1.636. RET_QUERY_DEC_ENCODING_START_TICK Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC_START_TICK																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.637. RET_QUERY_DEC_ENCODING_START_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ENC_START_TICK	R/W	Start tick of encoding slices of the picture	0x0

1.2.4.2.12.34. RET_QUERY_DEC_ENCODING_END_TICK (0x000001D0)

Table 1.638. RET_QUERY_DEC_ENCODING_END_TICK Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

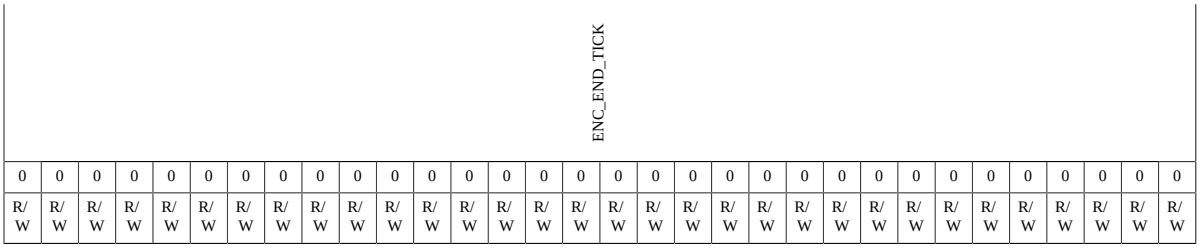


Table 1.639. RET_QUERY_DEC_ENCODING_END_TICK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ENC_END_TICK	R/W	End tick of encoding slices of the picture	0x0

1.2.4.2.12.35. RET_QUERY_ENC_ERR_INFO (0x000001D8)

Table 1.640. RET_QUERY_ENC_ERR_INFO Bit Assignment

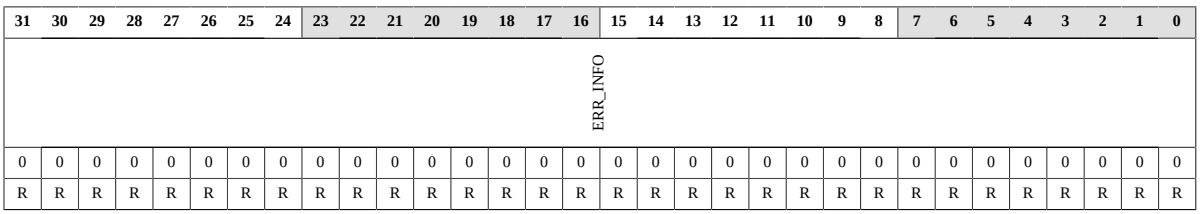


Table 1.641. RET_QUERY_ENC_ERR_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ERR_INFO	R	Error Information	0x0

1.2.4.2.12.36. RET_QUERY_ENC_SUCCESS (0x000001DC)

Query result

Table 1.642. RET_QUERY_ENC_SUCCESS Bit Assignment

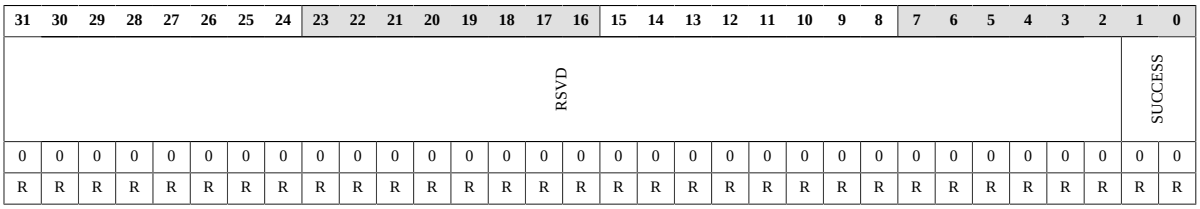


Table 1.643. RET_QUERY_ENC_SUCCESS Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1:0]	SUCCESS	R	It indicates that encoding result for the enqueued encode command(INIT_SEQ or ENC_PIC) 00: FAIL If the value is not "zero", it means "SUCCESS" 01: SUCCESS 10: SUCCESS_WITH_WARNING (success but there exist some warning) If it returns FAIL, please refer to RET_QUERY_ENC_ERR_INFO.	0x0

Bit	Name	Type	Function	Reset Value
			If it returns SUCCESS_WITH_WARNING, please refer to RET_QUERY_ENC_WARN_INFO.	

1.2.4.2.13. QUERY(GET_BS_WR_PTR) Command Parameter Registers

These are command I/O registers where Host processor can set arguments for the QUERY command with 6 of CMD_QUERY_OPTION or get return values.

1.2.4.2.13.1. CMD_QUERY_OPTION (0x00000104)

QUERY command option

Table 1.644. CMD_QUERY_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																										QUERY_OPTION					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.645. CMD_QUERY_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:6]	RSVD	R	Reserved	0x0
[5:0]	QUERY_OPTION	R/W	0x00: GET_VPU_INFO 0x02: GET_RESULT 0x03: UPDATE_DISP_IDC 0x04: GET_BW_RESULT 0x05: GET_BS_RD_PTR **0x06: GET_BS_WR_PTR**	0x0

1.2.4.2.13.2. RET_QUERY_ENC_BS_RD_PTR (0x00000114)

Table 1.646. RET_QUERY_ENC_BS_RD_PTR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QUERY_DEC_BS_RD_PTR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 1.647. RET_QUERY_ENC_BS_RD_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	QUERY_DEC_BS_RD_PTR	R	The start position of bistream buffer for the currnetly encoded picture.	0x0

1.2.4.2.13.3. RET_QUERY_ENC_BS_WR_PTR (0x00000118)

Table 1.648. RET_QUERY_ENC_BS_WR_PTR Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

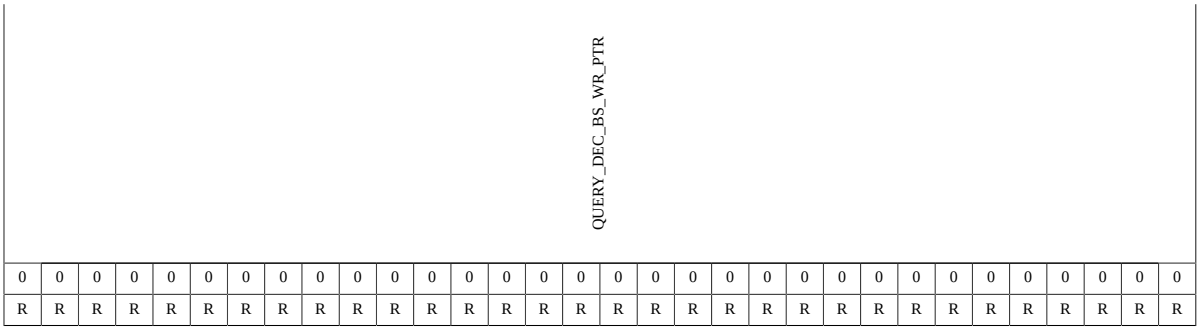


Table 1.649. RET_QUERY_ENC_BS_WR_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	QUERY_DEC_BS_WR_PTR	R	The end position of bistream buffer for the currnetly encoded picture	0x0

1.2.4.2.13.4. CMD_QUERY_ENC_REASON_SEL (0x0000011C)

Table 1.650. CMD_QUERY_ENC_REASON_SEL Bit Assignment

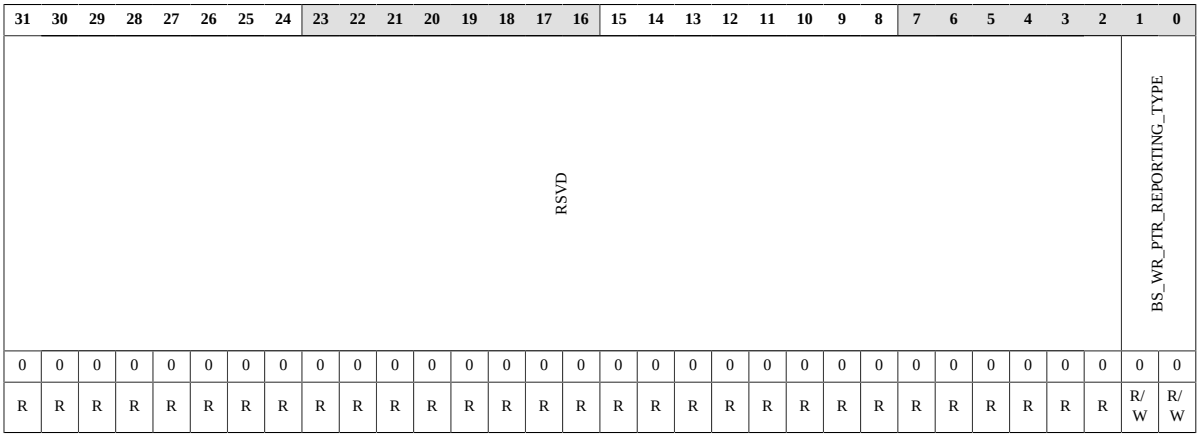


Table 1.651. CMD_QUERY_ENC_REASON_SEL Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R	Reserved	0x0
[1:0]	BS_WR_PTR_REPORTING_TYPE	R/W	0x00: It reports BS_RD_PTR / BS_WR_PTR at the time of ENC_PIC command done interrupt. 0x01: It reports BS_RD_PTR / BS_WR_PTR at the time of CPB_full occurance. 0x02: It reports BS_RD_PTR / BS_WR_PTR when low latency is used.	0x0

1.2.4.2.14. UPDATE_BS Command Parameter Registers for Encoder

These are command I/O registers where Host processor can set arguments for the UPDATE_BS command or get return values.

1.2.4.2.14.1. CMD_UPDATE_BS_OPTION (0x00000104)

UPDATE_BS command option

Table 1.652. CMD_UPDATE_BS_OPTION Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD																RSVD															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.653. CMD_UPDATE_BS_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:0]	RSVD	RW	Reserved	0x0

1.2.4.2.14.2. CMD_BS_START (0x00000118)

Bitstream buffer start address

Table 1.654. CMD_BS_START Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS_START_ADDR																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 1.655. CMD_BS_START Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_START_ADDR	R/W	Start address of bitstream buffer - line buffer mode only It should be aligned in bus width (128bit/16 byte), but we highly recommend to be aligned in 4KB	0x0

1.2.4.2.14.3. CMD_BS_SIZE (0x0000011C)

Bitstream buffer size

Table 1.656. CMD_BS_SIZE Bit Assignment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

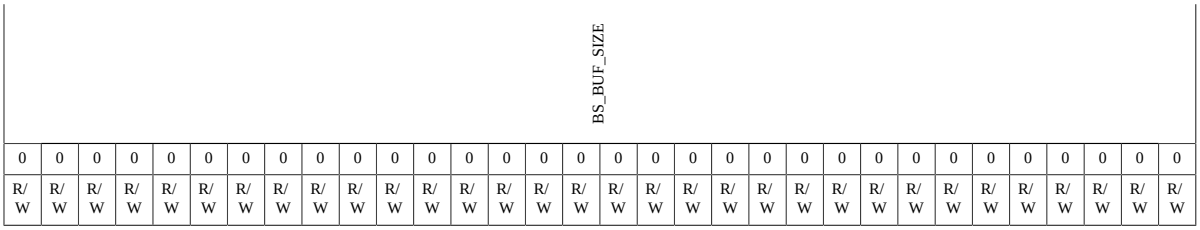


Table 1.657. CMD_BS_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_BUF_SIZE	R/W	Size of bistream buffer It should be aligned in bus width (128bit/16 byte), but we highly recommend to be aligned in 4KB	0x0

1.2.4.2.14.4. CMD_BS_OPTIONS (0x00000120)

Bistream buffer option

Table 1.658. CMD_BS_OPTIONS Bit Assignment

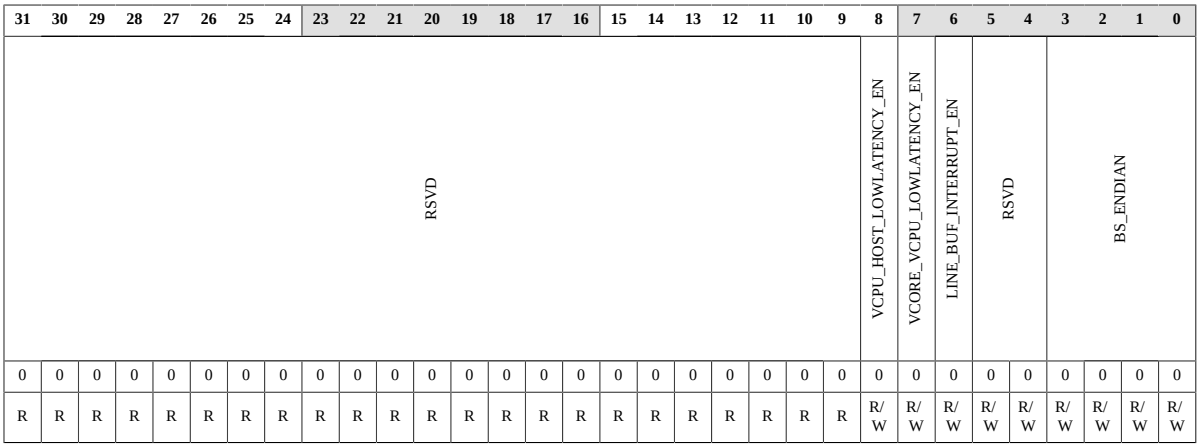


Table 1.659. CMD_BS_OPTIONS Field Description

Bit	Name	Type	Function	Reset Value
[31:9]	RSVD	R	Reserved	0x0
[8]	VCPU_HOST_LOWLATENCY_EN	R/W	It enables low latency mode between V-CPU and Host processor. If this is 1, V-CPU sends HOST an interrupt that a slice unit of bitstream is generated and ready to be transmitted. **Note** Low latency feature can be enabled only when the hardware supported NOTE: In case of SVAC ENC, VPU doesn't support low latency modes	0x0
[7]	VCORE_VCPU_LOWLATENCY_EN	R/W	It enables low latency mode between V-CORE and V-CPU. If this is 1, V-CORE sends V-CPU an interrupt that a slice of VLC stream is generated and ready to be transmitted to V-CPU for early start of entropy coding.	0x0

Bit	Name	Type	Function	Reset Value
			Note Low latency feature can be enabled only when the hardware supported NOTE: In case of SVAC ENC, VPU doesn't support low latency modes	
[6]	LINE_BUF_INTERRUPT_EN	R/W	TBD	0x0
[5:4]	RSVD	R/W	Reserved	0x0
[3:0]	BS_ENDIAN	R/W	Endianness of bitstream buffer	0x0

Chapter 2
APPLICATION INTERFACE

2.1. VPU API-based Control Mechanism

Host applications can control VPU at an API level through pre-defined API set. They can send a command with arguments, receive an interrupt indicating a requested operation has completed, or get a result of the command by using API functions as shown in *Figure 2.1, "SW control model of VPU from host application"*.

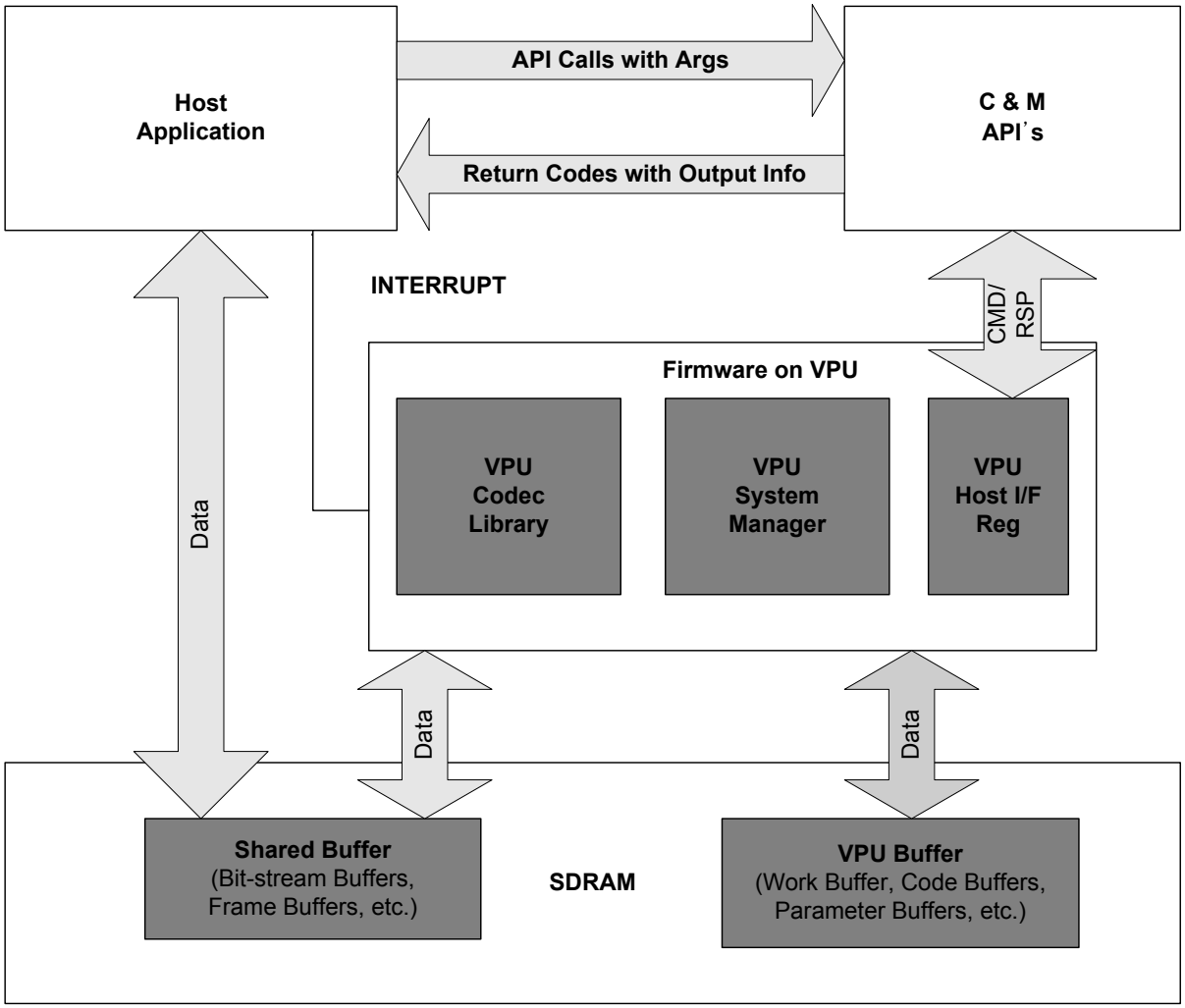


Figure 2.1. SW control model of VPU from host application

Each API definition includes the requested command as well as input and output data structure. The given command from API function is always written on a dedicated I/O register, and the input and output data structure are transmitted on a set of command I/O registers which contain input arguments and output results. So application programmers do not need to struggle with learning many of the host interface registers and their usage.

2.2. VPU API Reference Software

We provides customers with API reference software which is an implementation of codec application using VPU API functions. It helps application programmers develop their video applications by simply referring to or by porting it to fit their target CPU and OS.

There are five hierarchical layers in VPU API reference software, and *Figure 2.2, "API Reference Software Architecture"* shows the architectural layers of VPU API reference software.

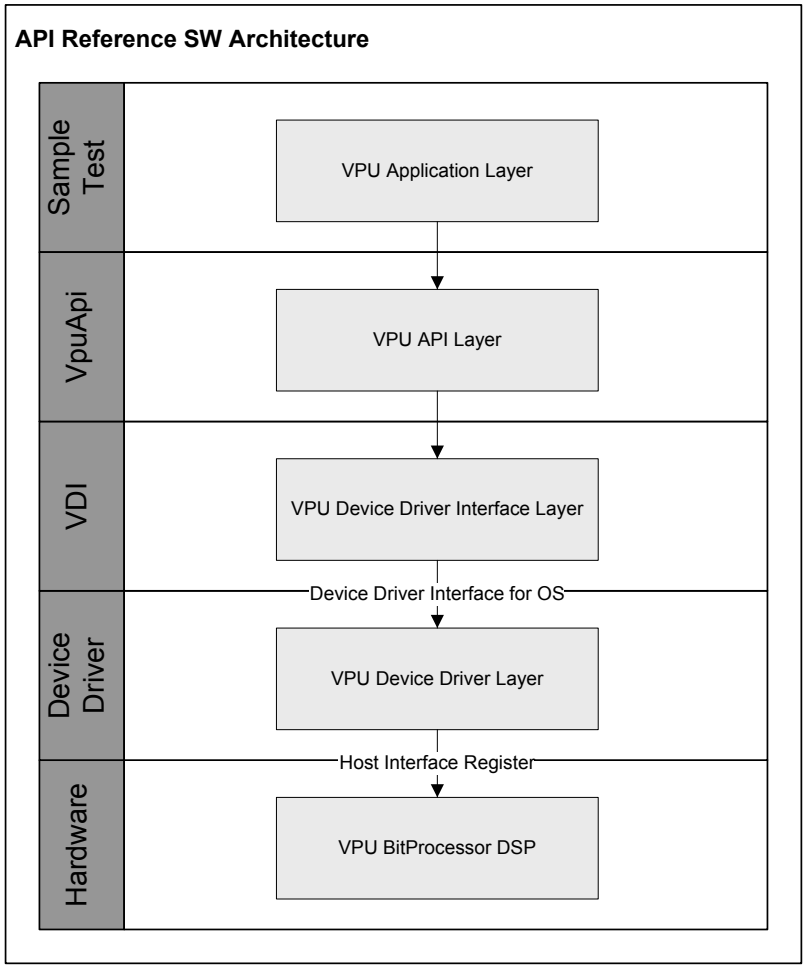


Figure 2.2. API Reference Software Architecture

- The VPU Application Layer is a kind of sample application stuff for decoding bitstream (packetized in general containers) and encoding image data by calling the VPU APIs.
- The VPU API Layer is the application-interface software stack to communicate with VPU hardware architecture. It can work on various OS platforms through VPU Device Driver Interface(VDI) that is able to get OS function calls.
- The VDI layer has some OS dependent codes for each OS that we support, and if there is a kernel mode in the OS, the VDI layer can communicate with the Device Driver Layer. Current API Reference Software implements three different types of VDI and their device drivers for Linux and NonOS and has a plan for extension and support for other OS.

2.2.1. Source Tree

Figure 2.3, "Source Tree of VPU API Reference Software" shows the source tree structure of the reference software package that we release.

<Root>/

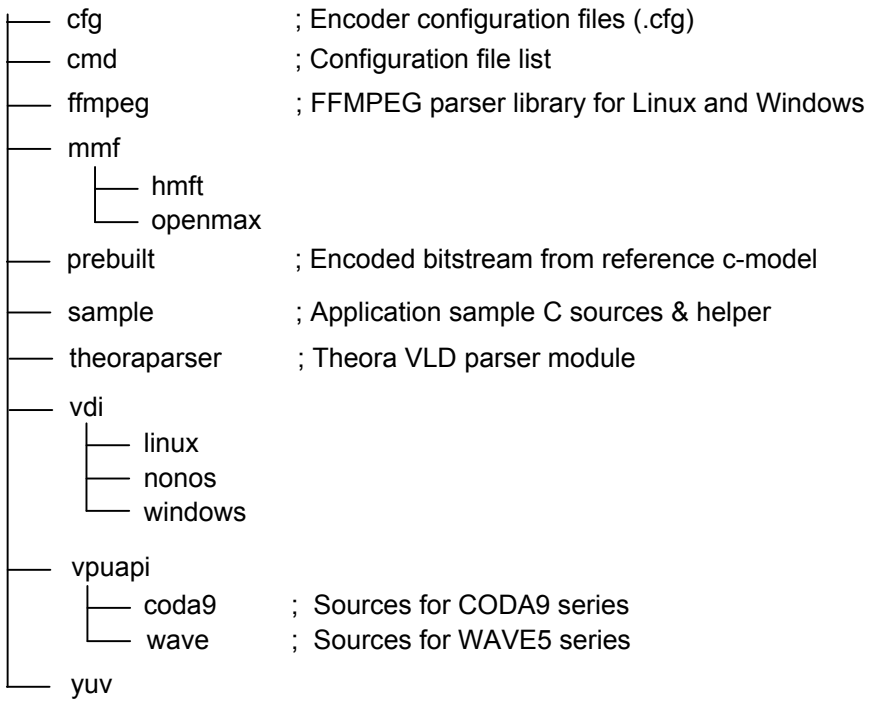


Figure 2.3. Source Tree of VPU API Reference Software

2.2.2. Architecture

2.2.2.1. Application Layer

As a top most layer of reference software, the VPU Application Layer has three functionalities. First, it contains video control sequences for decoding and displaying with given arguments. Second, it has a helper module that reads and writes a result from decoding and/or encoding and that interfaces with hardware resources through VDI module. And lastly, it also has user interface functions to communicate with application user through console menu.

To support these functions, the VPU Application Layer consists of Sample Test module, Parser module, VPUHelper module, and Mixer Module. *Figure 2.4, "Application Layer"* shows the modules of the Application Layer.

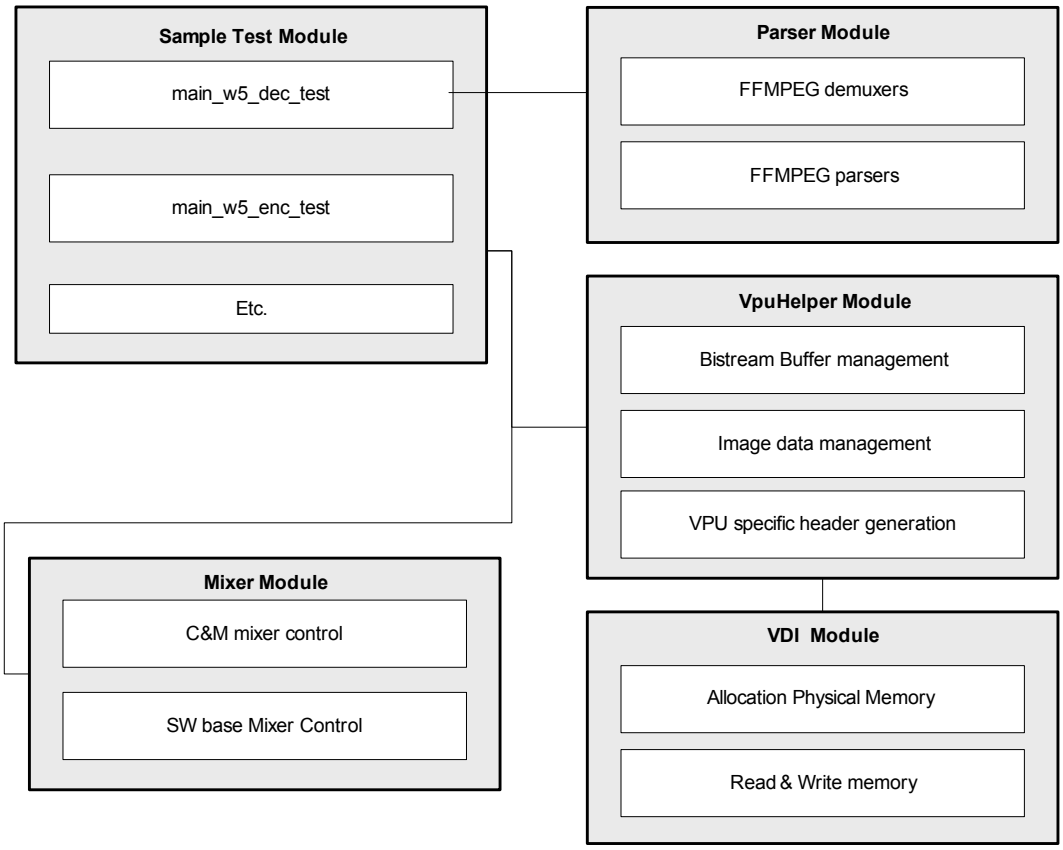


Figure 2.4. Application Layer

The relevant codes are in the `sample` directory , and they have simple functions such as Init, Open, SeqInit, Decode/Encode, Close, Deinit to control VPU hardware using VPU APIs, and also have system related codes that allocate decoder PP frame buffers, encoder source frame buffers, and bitstream buffer like user dependent memory. There is neither platform dependent code nor porting layer.

The following shortly describes responsibilities of each module and related files

- Sample Test module : main function in main_w5_enc_test.c or main_w5_dec_test.c
  - Start entry point
  - argc, argv parameter base
  - hevc_dec_test function with elementary stream
  - hevc_enc_test function with encoder option cfg file and user input encode option
  - MultiInstanceTest function : Multiple instance testing with multiple tasks that have different codec standards.
  - Calling VPU APIs and VDI to handle VPU

- **Parser module : ffmpeg open source library**
  - To make API reference software capable of extracting elementary stream from various multimedia containers.
- **VpuHelper module : sample/helper directory**
  - Helper functions for video codec interface
  - Bitstream directory : A code that reads and writes bitstream from/to the memory.
  - comparator : A function that compares result data between c-model and FPGA/SoC conformance test
  - display : display module
  - hm : Codes that reads out cfg files
  - misc : Other miscellaneous codes such as getopt for FPGA setting and md5
  - yuv : Codes that read and write YUV data from/to the memory
- **Mixer module : mixer.c**
  - Handle Chips&Media display module
  - Software based display support, including a yuv2rgb converter for display OS frame buffer

2.2.2.2. API Layer

The VPU API Layer is responsible for access to VPU hardware registers and direct control. This layer is composed of two modules, Main API module and Tilemap API module, which all are implemented in vpuapi.c and vpugdi.c. The task of Sample Test module from the upper Application Layer calls the VPU API functions in this API layer to control the VPU Hardware. There is neither platform dependent code nor porting layer.

Figure 2.5, "VPU API Layer" illustrates the modules of the VPU API Layer, hierarchy, and interactions.

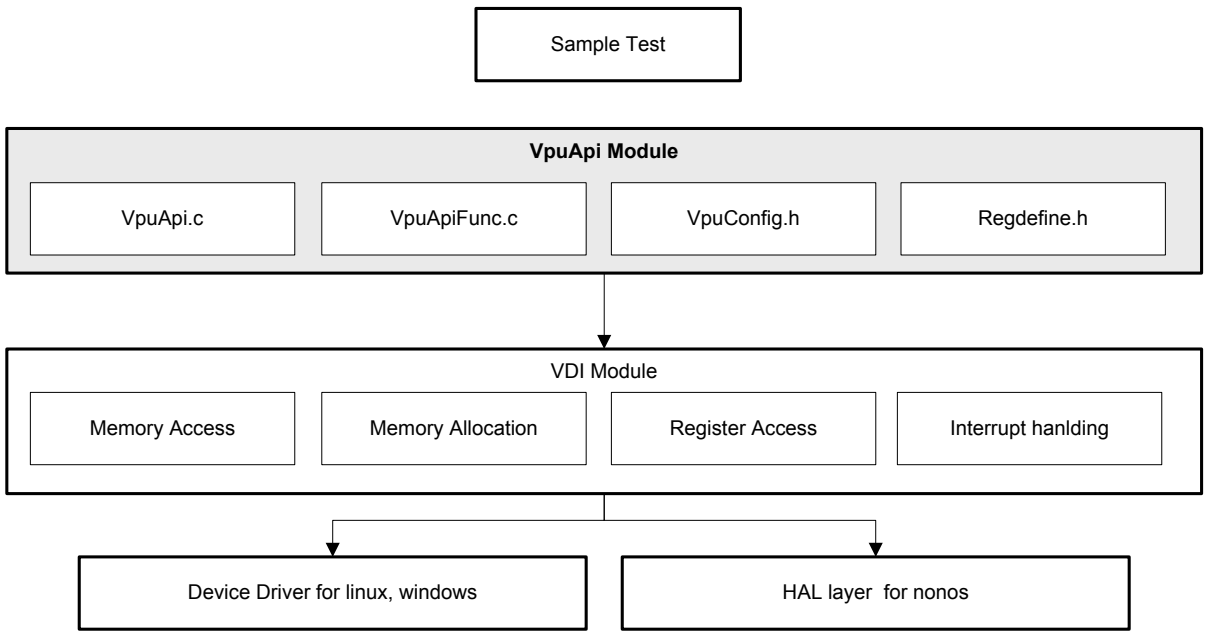


Figure 2.5. VPU API Layer

The following shortly describes responsibilities of each module and related files.

- **Main API module : VpuApi.c**
  - Decoder API function bodies to control the VPU Hardware and instance handle
  - All of the codec standard implementations share Host Interface on VPU hardware, so that these API functions can have simple architecture.
  - To support multiple instances, the opened instance handle pool resides under VpuApiFunc.c.
  - Some API functions access VPU Hardware registers directly through VDI functions.
  - Some API functions access VDI functions for allocating codec dependent memory and waiting for VPU interrupt
  - Some API functions jump to their sub-functions under VpuApiFunc.c to access the VPU hardware registers step by step.

2.2.2.3. VDI Layer

The VPU Device Driver Interface(VDI) can interface with OS layer or VPU directly. If the target system has a kernel mode such as Linux, VDI calls its device driver. For other OS like RTOS or Non OS platform, VDI can interface with the OS layer or VPU directly.

The relevant codes are in vdi directory and they have some porting layer (platform dependent codes) to integrate to a new target system. There is not any VPU dependency.

Figure 2.6, "VDI Layer" shows which components or function modules are in the VDI layer.

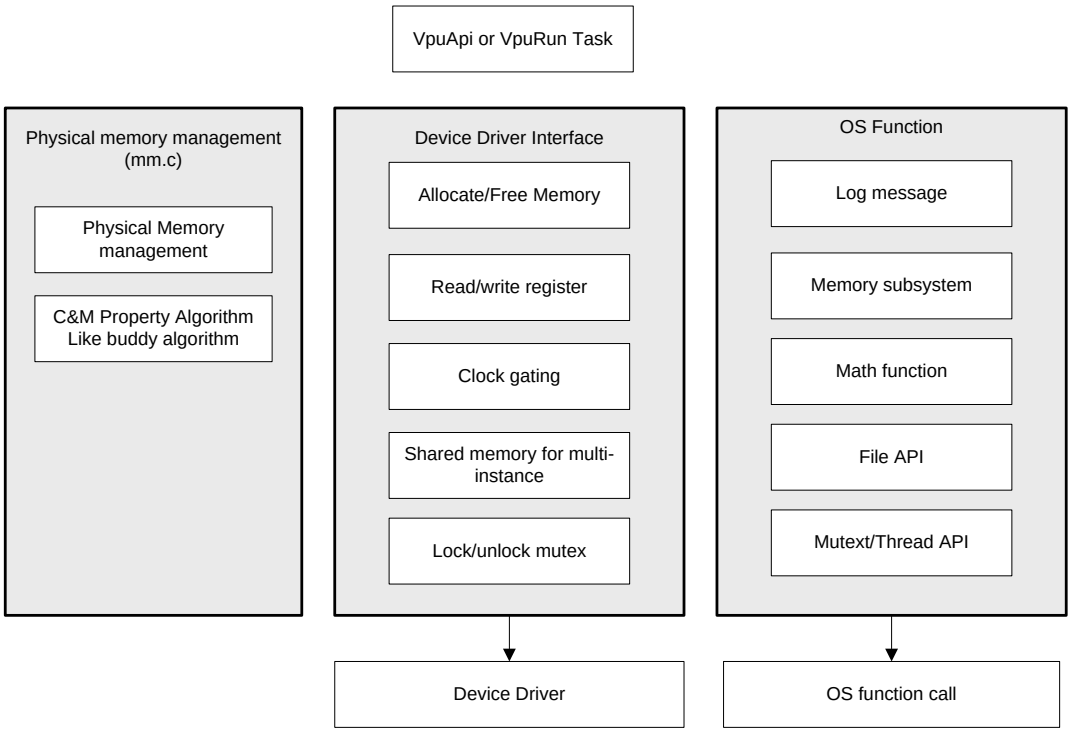


Figure 2.6. VDI Layer

The VDI Layer is taking charge of the following things.

- Read/Write VPU Register
- Read/Write physical memory
- Convert physical address to virtual address
- Allocate/Free memory
- Management of memory map for VPU in case not to use dynamic allocation, but to use system call
- Management of instances with shared memory in kernel mode
- SoC specified feature (HW reset, clock gating, interrupt)

2.2.2.4. Device Driver Layer

The Device Driver Layer accesses and handles VPU hardware, running in kernel mode if the OS has a framework for device driver. The relevant codes reside in VDI/[OS]/driver directory. It allocates physical buffer by using OS function calls, handles an interrupt, manages shared memory that is used for vpuapi handles for multi-process application.

2.2.3. Supported Operating Systems

2.2.3.1. Linux

VPU Reference Software is ported to Embedded Linux kernel version 2.6.35, with including OpenMAX-IL component, OMX core, and Gstreamer plugin. The relevant codes are in VDI/linux folder.

- Build Environment
  - Toolchain : Sourcery CodeBench Lite 2011.09-65 for GNU/Linux
  - makefile in root folder : BUILD_CONFIGURATION variable should be Debug_Embedded_Linux. Also configure CC, CXX, and AR variables in case that cross compiler needs to be changed.
- Verification platform
  - ARM + FPGA platform board of SOCLE inc.

2.2.3.2. Android

VPU Reference Software is ported to Android version 2.3.4, with including OpenMAX-IL component, OMX core, and Stagefright plugin. The relevant codes are in VDI/linux folder.

- Build Environment
  - Toolchain : Android Built-in Toolchain
  - Android.mk in root folder.
- Verification platform
  - ARM + FPGA platform board of SOCLE inc.

2.2.3.3. Non OS

VPU Reference Software is ported to ARM core base without a special OS. The relevant codes are in VDI/nonos folder.

- Build Environment

  - Sourcery CodeBench Lite 2011.09-65 for ARM EABI
  - makefile in root folder : BUILD_CONFIGURATION variable should be `Debug NonOS`.
- Verification Platform
  - MQX RTOS for certain ARCH platform
  - UCOS RTOS for certain ARM platform

2.3. Porting to Target System

2.3.1. Identifying Build Tool(c - compiler)

Example 2.1. config.h

```
-------------------------------------------------------------------------------
#       define PLATFORM_NON_OS

#  error "Unknown compiler."
-------------------------------------------------------------------------------
```

Application should choose their target OS among PLATFORM_WIN32, PLATFORM_LINUX, PLATFORM_NON_OS

- PLATFORM_LINUX : when target platform is either embedded Linux or Android
- PLATFORM_WIN32 : when target platform is Windows Embedded Compact 7 or Windows8
- PLATFORM_QNX : when target platform is Unix
- PLATFORM_NON_OS : when not in use of OS

For any other OS that is not defined here, new definition and platform dependent codes for that OS might be added.

2.3.2. Porting VDI

2.3.2.1. Creating a New VDI Folder

The first thing to port VDI is creating a new VDI folder and adapting the vdi.c and vdi_osal.c file to a new target system.

2.3.2.2. vdi Function Prototype

Example 2.2. vdi.h

```
-------------------------------------------------------------------------------
int vdi_probe(unsigned long core_idx);
int vdi_init(unsigned long core_idx);
int vdi_release(unsigned long core_idx);    //this function may be called only
                                              at system off.

vpu_instance_pool_t * vdi_get_instance_pool(unsigned long core_idx);
int vdi_get_common_memory(unsigned long core_idx, vpu_buffer_t *vb);
int vdi_allocate_dma_memory(unsigned long core_idx, vpu_buffer_t *vb);
void vdi_free_dma_memory(unsigned long core_idx, vpu_buffer_t *vb);
int vdi_get_sram_memory(unsigned long core_idx, vpu_buffer_t *vb);
int vdi_wait_interrupt(unsigned long core_idx, int timeout, unsigned long
```

```
addr_bit_int_reason);
int vdi_hw_reset(unsigned long core_idx);
int vdi_set_clock_gate(unsigned long core_idx, int enable);
int vdi_get_clock_gate(unsigned long core_idx);
int vdi_get_instance_num(unsigned long core_idx);
void vdi_write_register(unsigned long core_idx, unsigned long addr, unsigned int data);
unsigned long vdi_read_register(unsigned long core_idx, unsigned long addr);
int vdi_write_memory(unsigned long core_idx, unsigned long addr,
unsigned char *data, int len, int endian);
int vdi_read_memory(unsigned long core_idx, unsigned long addr,
unsigned char *data, int len, int endian);
int vdi_lock(unsigned long core_idx);
void vdi_unlock(unsigned long core_idx);
int vdi_disp_lock(unsigned long core_idx);
void vdi_disp_unlock(unsigned long core_idx);
int vdi_wait_vpu_busy(unsigned long coreIdx, int timeout, unsigned long addr_bit_busy_flag);
void vdi_log(unsigned long coreIdx, int cmd, int step);
-------------------------------------------------------------------------------------
```

2.3.2.3. Implementation of vdi.c

#define VDI_SYSTEM_ENDIAN, VDI_LITTLE_ENDIAN

Sets an endian mode according to SoC's bus endian. For example, SoC uses AXI bus of ARM processor, VDI_LITTLE_ENDIAN should be defined.

#define VPU_BIT_REG_BASE 0x10000000

Sets the base address for VPU hardware register in SoC's memory map. When there is a device driver on the target platform, the device driver does set this address and so application does not need to set.

#define VDI_SRAM_BASE_ADDR 0x00

Sets the base address for VPU Secondary AXI bus (SRAM).

Note Set VDI_DRAM_PHYSICAL_BASE instead when you use VDI for Linux or Windows.

#define VDI_SRAM_SIZE 0x25000

Sets the size of VPU Secondary AXI bus (SRAM).

#define VDI_DRAM_PHYSICAL_BASE 0x00

Assigns the base address of memory that VPU uses. When there is a device driver on the target platform, application should make the device driver get the base address through the IOCTRL, VDI_IOCTL_GET_RESERVED_VIDEO_MEMORY_INFO.

#define VDI_DRAM_PHYSICAL_SIZE (1024*1024*1024)

Sets total size of memory that VPU uses

Note When you use VDI for Linux or Windows, set VPU_INIT_VIDEO_MEMORY_SIZE_IN_BYTE instead which is found vdi/linux(or windows)/driver/vpu.c.

#define SUPPORT_MULTI_CORE_IN_ONE_DRIVER

Enable this when VPU cores are in a same SOC using one DRAM. Comment out this code when VPU cores are in different SOCs using their own DRAM.

vdi_init()

This function creats a new vdi handle. For the OS with device driver, this function loads the device driver. It creates VPU instance mutexes for each OS and display lock mutex handles.

vdi_hw_reset()

This function is for VPU hardware reset. It should include some codes to issue a reset signal for VPU according to SoC environment.

vdi_lock(), vdi_unlock()
In multi-thread(multi-task) environment, a pair of these functions prevent VPU as a critical section from being concurrently accessed by threads. Add a mutex lock/unlock function according to target system. For example, application can use `pthread_mutex_lock()` and `pthread_mutex_unlock()` function when its target system is Linux.

vdi_write_register(), vdi_read_register()
These functions write to and read from Host Interface Register of VPU. Register access function is required to implement accoring to target system. It is supposed to direct access to VPU register address with volatile keyword as a default setting.

vdi_write_memory(), vdi_read_memory()
This function transfers data between CPU memory and VPU memory. `vdi_write_register()` copies CPU memory data (buffer pointer as an input argument) to the VPU target address. Vice versa `vdi_read_register()` copies data from VPU memory to CPU memory. As default a `memcpy()` function is used for this, but it might be possible to replace with a system dma copy function.

vdi_set_clock_gate(), vdi_get_clock_gate()
This function enables VPU clock gating. Application should implement a function to gate or ungate the VPU clock according to SoC environment. It is an option for low power, not mandatory to implement.

vdi_get_sram_memory()
This function returns the address of VPU secondary memory. Application does not need to port it, but simply assign the base address to VDI_SRAM_BASE_ADDR.

vdi_get_common_memory()
This function returns common DRAM memory address that is used by VPU. Porting is not desired.

vdi_wait_interrupt
This function waits for an interrupt. If SoC supports an interrupt, application should add an interrupt waiting function for the target OS. We provide a polling function that reads out VPU status register periodically for the SoC not using an interrupt, and application should add the Sleep(1) code in it according to your target system because timeout is calculated in unit of 1ms.

2.3.2.4. Implementation of vdi_osal.c

LogMsg function()
This function sets print of debug string such as UART print.

osal_memcpy(), osal_memset(), osal_malloc(), osal_free()
This function manages virtual memory of OS. VDI controls the physical memory used for VPU DMA.

osal_fxxx()
This function is for file processing. Application(VPURUN) code load or save bistream and image output in a file format, so there is need for application to work with file system .

link system library
Since VDI calls system related functions (malloc, memcpy, and so forth), libc libaries for the target OS should be prepared and linked.

2.3.3. Configuration of VPU

2.3.3.1. VPUAPI/vpuconfig.h

MAX_INST_HANDLE_SIZE
It is the size of variable holding information of instances that are managed inside driver. It must not be changed.

MAX_NUM_INSTANCE
Configures the number of instance to run if application wants to operate more than one instance. Instances could be set as many as possible within DRAM size available.

MAX_NUM_VPU_CORE
Sets the number of VPU cores if target SoC does have more than one VPU core to enable parallel processing or multi-channel.

VPU_BUSY_CHECK_TIMEOUT
Sets waiting time until a certain command is executed. This is millisecond unit and if there is not any response from it, VPU returns the error code, RETCODE_VPU_RESPONSE_TIMEOUT.

VPU_FRAME_ENDIAN
Sets an endian mode for frame buffer(image) data. i.e. VDI_128BIT_LITTLE_ENDIAN is set when ARM processor and AXI Bus is used.

VPU_STREAM_ENDIAN
Sets an endian mode for bistream buffer data VDI_128BIT_LITTLE_ENDIAN is set when ARM processor and AXI Bus is used.

CBCR_INTERLEAVE
Sets a CBCR interleave mode
- 1 : CBCR interleaved
- 0 : CBCR separated

i.e. Set this to 1 when FOURCC value is NV12. Or set this to 0 when FOURCC value is YV12.

NV21_ENABLE
Specifies the chroma interleave format.
- 1 : NV21
- 0 : NV12

SIZE_COMMON
It is the total size of code memory and stack memory which all are used by VPU.

WAVE5_MAX_CODE_BUF_SIZE
It is the size of firmware binary file. 1MB or less should be allocated for this code and stack memory.

WAVE521DEC_WORKBUF_SIZE/WAVE521ENC_WORKBUF_SIZE
It is the size of work buffer where some variables for running VPU are saved. More than 512KB of memory region should be allocated for this.

MAX_NUM_VCORE
It is the number of VCE core inside VPU. It should set to 1.

2.3.4. Porting Device Driver

When user mode and kernel mode are separated on the OS such as Linux, it is expected to do porting the VDI part to fit into a new target OS. Application should implement a device driver for framework of the target OS.

2.3.4.1. IO Control Codes

Porting IOCTL Codes for Linux driver

```
VDI_IOCTL_MAGIC  'V'

VDI_IOCTL_WAIT_INTERRUPT                    IO(VDI_IOCTL_MAGIC, 2)
```

```
VDI_IOCTL_SET_CLOCK_GATE                    _IO(VDI_IOCTL_MAGIC, 3)
VDI_IOCTL_RESET                             _IO(VDI_IOCTL_MAGIC, 4)
VDI_IOCTL_GET_INSTANCE_POOL                 _IO(VDI_IOCTL_MAGIC, 5)
VDI_IOCTL_GET_RESERVED_VIDEO_MEMORY_INFO    _IO(VDI_IOCTL_MAGIC, 8)
```

VDI_IOCTL_WAIT_INTERRUPT
A waiting function based on interrupt scheduling for the device driver framework.

VDI_IOCTL_SET_CLOCK_GATE
Clock gating on/off

VDI_IOCTL_RESET
HW reset signal on/off

VDI_IOCTL_GET_INSTANCE_POOL
Returns shared memory which is used by VDI and VPU API This memory should always be returned with same region.

VDI_IOCTL_GET_RESERVED_VIDEO_MEMORY_INFO
Allocates the entire size of memory that is used for VPU hardware and returns the address. Video processing demands considerable of memory space and most of the memory allocations should be done at booting time, which is also dependent on OS. Therefore device driver actually allocates memory for VPU at booting time, and VDI returns the memory information by the VDI_IOCTL_GET_RESERVED_VIDEO_MEMORY_INFO io-control.

Note For an example of making a new device driver, please refer to sample driver codes in vdi/linux/driver folder.

2.3.4.2. Device Driver Configuration

#define VPU_SUPPORT_CLOCK_CONTROL
Enable Clock Gating control by using clock library in Linux kernel.

#define VPU_SUPPORT_ISR
Enable the use of interrupt service routine.

#define VPU_SUPPORT_PLATFORM_DRIVER_REGISTER
Get a base address and IRQ number from platform driver library.

#define VPU_INIT_VIDEO_MEMORY_SIZE_IN_BYTE
Set the memory size for memoy allocation for running VPU. It must be larger than REQUIRED_VPU_MEMORY_SIZE that is calculated by vpuconfig.h.

#define VPU_SUPPORT_RESERVED_VIDEO_MEMORY
Enable the use of reserved memory region for VPU memory. When this is disabled, application should allocate memory by calling kernel API for device driver's non-cache/continuous physical memory allocation. For example, Linux driver uses a dma_alloc_coherent function.

#define VDI_DRAM_PHYSICAL_BASE
Set the physical base address of reserved memory if VPU_SUPPORT_RESERVED_VIDEO_MEMORY is enabled.

2.4. Verification

For information on how to run sample decode/encode test using Chips&Media API reference software, please refer to the *Chapter 4. Software Verification Environment* of *Verification Guide*.

Chapter 3
HOW TO CONTROL VPU

3.1. VPU Initialization

When host processor turns on VPU for the first time, host processor needs to follow the steps below to activate VPU, which is called an initialization process. All these manual procedures including VPU hardware reset, firmware load, interrupt enable, and VPU start can be replaced with simply calling a single API function `VPU_Init()` that we provide.

1. Set VPU_PO_CONF register to 0.
2. Reset all hardware blocks by setting VPU_RESET_REQ register to 0x7FF_FFFF.
3. Wait until VPU_RESET_STATUS register becomes 0.
4. Set VPU_RESET_REQ register to 0.
5. After the reset, now VPU_REMAP_CORE_START register has 0.
6. Place the binary file of firmware in the SDRAM where is accessible by VPU.
   a. Set ADDR_CODE_BASE register to SDRAM address where the firmware is placed. ADDR_CODE_BASE must be aligned in 4KB boundary.
   b. Set CODE_SIZE to the size of firmware.
   c. Set CODE_PARAM with endian. This should be same with the firmware's endian.
7. VPU starts booting from 0x0 of virtual address. Therefore, the code region should be remapped to virtual address for VPU,
   a. Set VPU_REMAP_CTRL register. (set GLOB_EN flag to 1 for global setting such as Endianness and enter the values of ENDIAN.)
   b. Set VPU_REMAP_VADDR to 0x0 (Code Section always starts from 0)
   c. Set VPU_REMAP_PADDR with ADDR_CODE_BASE.
8. Set HW_OPTION register for activating hardware options such as UART or debug option. Generally it is 0.
9. Set VPU_VINT_ENABLE register to enable an initial interrupt if neccessary.
10. Set VPU_FIO_DATA using the programmable AXI ID. It is optional. (default: 0)

    [31:28] Processor AXI ID (not supported in one AXI environment)
    [27:24] PRP AXI ID (not supported in one AXI environment, encoder only)
    [23:20] FBD_Y AXI ID (not supported in one AXI environment)
    [19:16] FBC_Y AXI ID (not supported in one AXI environment)
    [15:12] FBD_C AXI ID (not supported in one AXI environment)
    [11:08] FBC_C AXI ID (not supported in one AXI environment)
    [7:4] Primary AXI ID
    [3:0] Secondary AXI ID

11.Set VPU_FIO_CTRL_ADDR with the programmable AXI ID register address which is 0xFE0C. It is optional.

12.Write VPU_BUSY_STATUS register to 1.

13.Set COMMAND(0x100) register to 1 for INIT_VPU command.

14.Finally set VPU_REMAP_CORE_START to 1 to start VPU.

Detailed information about each initialization step and some programming tips are presented in the following sub-sections.

Note | A total code size of firmware could be varied according to VPU configuration and customization.

3.1.1. Version Check of VPU hardware and Firmware

Application can check the version information of VPU hardware and firmware by using GET_FW_VERSION command. The version number of firmware is presented by a 32 bit value.

- Revision[31:0] - F/W revision number

The dedicated command is used for this version check, and also this is supported by calling `VPU_GetVersionInfo()` function after initialization.

3.1.2. Data Buffer Management

VPU requires a certain amount of SDRAM space for decoding or encoding operation. This dedicated memory space for VPU includes a code buffer, a working buffer, and a temp buffer.

- A code buffer is a memory region where firmware binary is stored. Host processor should set the base address of code buffer so that VPU can start boot-up from the address. VPU has one code buffer regardless of number of instances opened.
- A working buffer is a memory region where the parameters and information of a sequence are saved. An instance has one working buffer.
- A temp buffer is a memory region that holds temporarily required information for performing actual decoding process such as intra prediction and deblocking filter. Picture size affects the size of temp buffer. Part of temp buffer might be connected to on-chip SRAM through secondary AXI bus to help reducing external bandwidth. VPU has its own temp buffer that can be shared by all the instances opened. However, in multi-VPU enviroment there are temp buffers as many as the number of cores.

Each buffer size might vary according to codec standard and picture size of stream that an instance uses. Thus the minimum reqired sizes of the buffers except code buffer are returned by INIT_SEQ command respectively. The size of code buffer is the same as binary of the firmware.

Basically all of the buffers should be aligned with bus width (128-bit/16-byte), except for some buffers in 4K alignment.

Allocation of Buffers

Host processor can assign a code buffer simply by calling `VPU_Init()` that automatically sets and manages individual buffers at once through VDI module.

A working buffer is assigned whenever an instance opens and it is as large as the size defined in `VPU_DecOpen()` or `VPU_EncOpen()`. Through the VPUAPI functions, the address of each buffer is set to the dedicated registers of Host Interface register.

A temp buffer is allocated when calling VPU_DecIssueSeqInit() for decoder or VPU_EncGetInitialInfo() for encoder with the size as much as VPUAPI requires inside.

3.1.3. Bitstream Buffer Management

A bitstream buffer is a memory region that stores coded picture data which is known as bitstream. Host processor and VPU share the bitstream buffer.

There are two bitstream buffer modes, a linear buffer mode and a ring buffer mode. A linear buffer mode allocates bitstream buffer dynamically with start and end region of bitstream in byte unit. A ring buffer mode reserves a fixed size of memory region. In this mode, bitstream is filled and read from the buffer with a read pointer and a write pointer as known as circular buffer scheme.

Host processor should set BS_START_ADDR and BS_SIZE register in alignment with 128-bit bus width, which is to inform VPU of the start address of bitstream buffer and its size. With setting the BS_START_ADDR and BS_SIZE register dynamically for every ENC_PIC command, bitstream buffer can run in linear buffer mode.

Host processor can set bitstream buffer parameters such as endianness, way of bitstream pumping or handling a bitstream shortage case on BS_PARAM register. VPU does not allow change of bitstream buffer parameters in a same instance. In other words, different parameters might be set for another instance.

However, host processor can update EXPLICIT_END flag of BS_OPTION register anytime in the same instance. EXPLICIT_END is an option that can force to decode a frame even if out of bitstream happens in the buffer while decoding.

3.1.3.1. Allocation of Bitstream Buffer

In ring buffer mode, host processor should allocate bitstream buffer in advance. Bitstream buffer should be aligned to memory bus width. For example, if a host processor uses a 128-bit bus, bitstream should be aligned to a 128-bit boundary. Thus, host processor should set carefully BS_START_ADDR and BS_SIZE register to meet this requirement.

Even though there are no particular restrictions on bitstream buffer size, we recommend to assign bitstream buffer at least larger than one coded picture data which is affected by bitrate.

3.1.3.2. Pointers for Reading Bitstream Buffer (Encoder)

To access encoded bitstream, there are two pointers to bitstream buffer, a read pointer (BS_RD_PTR) and a write pointer (BS_WR_PTR). When it comes to controlling these pointers, host processor can basically manipulate both pointers before sending a command to VPU. However, host processor is not allowed to control pointers while encoding.

BS_RD_PTR indicates the start of stream, and BS_WR_PTR indicates the end of stream. Once picture encoding is completed, host processor can get bitstream from BS_RD_PTR to the encoded byte size (RET_ENC_PIC_BYTE) or to BS_WR_PTR.

3.1.3.3. Pointers for Bitstream Pumping Operation (Decoder)

There are two pointers - a read pointer(BS_RD_PTR) where VPU is reading stream data from the buffer and a write pointer(BS_WR_PTR) where the buffer is being filled up. This scheme is preferred in packet-based video communication and streaming applications such as broadcasting or video conference. In this kind of packet streaming based on a ring-buffer, a read pointer and write pointer are automatically wrapping around at the boundaries of the buffer.

When it comes to control of these pointers, host processor can basically manipulate both pointers before sending a command to VPU. However, host processor cannot change a read pointer while VPU is decoding and can manipulate a write pointer after feeding bitstream to bitstream buffer.

When applications are going to feed a new chunk of bitstream, they need to check if there is available space to write in bitstream buffer, which can be computed simply with a read pointer, a write pointer and a buffer size. The API

VPU_DecGetBitStreamBuffer() is a dedicated function for that, informing applications of location of read pointer and write pointer and available space in bitstream buffer. With the return value of this API function, applications can download a new chunk of bitstream whose size is smaller than available buffer space to bitstream buffer. There is another API, VPU_DecUpdateBitStreamBuffer() that allows applications to get informed the amount of bits transferred into bitstream buffer (BS_WR_PTR).

Host processor can use API functions to manipulate BS_RD_PTR and BS_WR_PTR.

- VPU_DecSetRdPtr() updates BS_RD_PTR
- VPU_DecUpdateBitstreamBuffer() updates BS_WR_PTR

3.1.3.4. Bitstream Handling Modes (Decoder)

When VPU starts decoding, VPU executes prescan to find if a complete NAL exists in bitstream buffer and loads 4KB chunk of bitstream from the buffer. If it turns out there is enough NALs to decode one frame, VPU does not load any more bitstream.

When VPU fails to decode from bitstream shortage, VPU behaves according to either of two bitstream handling modes, interrupt mode and PicEnd mode.

- In the interrupt mode, VPU does not do anything and waits until host processor fills bitstream in the buffer.
- In the PicEnd mode, VPU takes action according to the given BS_SHORTAGE_OPTION flag of BS_PARAM register. It is either concealment or error report.

Host processor can select either Interrupt mode or PicEnd mode in the EXPLICIT_END flag of BS_OPTION register.

Bitstream handling mode should be set before issuing DEC_PIC command. Host processor can change from interrupt mode to PicEnd mode while VPU is running. However, for switch back to the interrupt mode, host processor should wait until VPU has completed on-going DEC_PIC command.

One more thing that host processor needs to be careful is use of BS_WR_PTR in PicEnd mode. Host processor must not update BS_WR_PTR until one picture decoding is finished. WR_PTR should be static while decoding for safe operation.

3.1.3.4.1. Interrupt Mode

Interrupt mode is a basic mode for handling bitstream in VPU. If remaining bitstream data in the bitstream buffer is less than 512 bytes, VPU sends an interrupt to host processor to ask for more bitstream. In this case, the interrupt reason is set as bitstream buffer empty that is 14th bit in BIT_INT_REASON. After receiving the interrupt, host processor should fill bitstream to the bitstream buffer with new coded picture data or set EXPLICIT_END flag. Meanwhile, VPU waits for more bitstream to be fed up or the EXPLICIT_END flag to be updated if there exist not enough bitstream to decode.

Note We recommend that host processor should feed bistream larger than 1024 bytes into the bitstream buffer when it is right after SEQ_INIT command for safe decoder operation.

NAL level pumping

In the interrupt mode, VPU tries to decode to the almost end of bitstream buffer (WR_PTR), but last three bytes or less in the bitstream buffer. Those last bytes are intended not to be consumed during decoding, because they might be part of start code for the next NAL, or because they might not be filled into the offset in the CABAC (CABAC needs more than 4 byte chunk.)

To overcome this problem, VPU has NAL level pumping mode. The size of a NAL unit can be determined in host parser by checking startcode of the next NAL unit , by checking STOP pattern, or by using size information in

container header. In interrupt mode, VPU hard to detect the end of NAL thus some of bytes cannot be decoded. In NAL level pumping, VPU assume the position of BS_WR_PTR is always the end of a NAL unit, thus, it can consume all the byte in the bitstream buffer if host processor can guarantee. User can enable this mode by setting NAL_END and EXPLICIT_END flags.

This mode is the simplest and efficient way of bitstream management, but it has some weak points as follows.

- If streaming coded picture data for a frame is much slower than consuming, it might lead unacceptable latency in instance switch (multi-instance use-case), because VPU switches a current instance with another only if the current instance finishes its decoding.

To handle these situations, host processor can do mode-switch into PicEnd mode, which continues decoding with mere small bitstream.

3.1.3.4.2. PicEnd Mode

In this mode, VPU decodes until the end of of bitstream that host processor fills in. After the decoding, VPU might encounter either of the following two cases:

- If it was end of bitstream for a picture and VPU finishes decoding a whole frame successfully, VPU returns the result of picture done as it normally does.
- If it was not enough to complete decoding a frame or sequence header, VPU assumes that this is bitstream shortage case, and it performs predefined operation by the BS_SHORTAGE_OPTION@BS_OPTION

Assume Error on bitstream shortage

In this option, VPU assumes error in the end of stream and try to conceal error for the remaining pixels in the picture. After concealment, VPU return RET_SUCCESS as 0x2 (Success with warning) and write the FAIL_REASON. This mode is very useful in low-latency application and with small size of bitstream buffer.

File-Play Emulation using PicEnd mode

The PicEnd mode can emulate file play operation by designating where a certain file data begins and ends with BIT_RD_PTR and BIT_WR_PTR respectively. VPU can decode a stored coded picture data directly from BIT_RD_PTR to BIT_WR_PTR, without copying to bitstream buffer for decoding. To work this way, BIT_RD_PTR and BIT_WR_PTR should stay in bitstream buffer region. In other words, host processor should set bitstream buffer large enough to hold many files. Generally, host processor sets almost all of external memory as bitstream buffer for file-play emulation using PicEnd mode.

Report Error on bitstream shortage without addition handling

In this mode, VPU returns RET_SUCCESS as 1 immediately without any additional operation. In this case, host processor can drop the frame or conceal in host side.

3.1.3.5. Considering Multiple Instances

With multi-instance feature, host processor has to manage its own list of BIT_RD_PTR and BIT_WR_PTR as many as instances are created with RunIndexes in order to identify each instance. A BIT processor sets the external SDRAM Bitstream read address with current RunIndex to the BIT_RD_PTR register, so that host processor can get informed how amount of bitstream has been read and decoded for a specific instance.

3.1.4. Interrupt Signaling Management

In order to achieve maximum efficiency in VPU control, VPU basically provides interrupt signaling for completion of a requested operation as well as stream buffer empty/full. But for some commands returning quickly, interrupt signaling is not provided because interrupt signaling is not helpful in that case.

Currently, C&M provides interrupt signaling for the following commands:

- VPU_INIT : VPU processor initialization has been completed after setting VPU_REMAP_CORE_START by 1.
- WAKEUP_VPU : WAKEUP_VPU command has been completed.
- SLEEP_VPU : SLEE_VPU command has been completed.
- CREATE_INST : Instance creation has been completed
- FLUSH_INST : Forcing decoder to flush bitstream buffer and frame buffers has been completed.
- DESTROY_INST : VPU sequence termination has been completed.
- SET_FB : Registration of frame buffer has been completed.
- INIT_SEQ (for decoder) / SET_PARAM (for encoder): Sequence initialization/ has been completed.
- DEC_PIC/ENC_PIC : VPU picture processing has been completed.
- QUERY : Retrieving command result or hardware status has been completed.
- UPDATE_BS : Updating bistream buffer has been completed.

Each interrupt could be easily enabled or disabled by writing 0 or 1 to the corresponding bit field of Interrupt Enable Register. And when an interrupt is signaled, applications can check the source of interrupt by checking the value of Interrupt Reason Register.

All these kinds of interrupt signaling could be replaced by a polling scheme of reading VPU_BUSY_STATUS register in VPU, when interrupt signaling is not easily applicable.

3.2. Encoder Control

3.2.1. Overall Encoder Sequence

Figure 3.1, "Encoder Control Flow with APIs" gives a brief summary of encoder control flow that shows the relation between encoder control flow and API function. And it also represents short description of each stage.

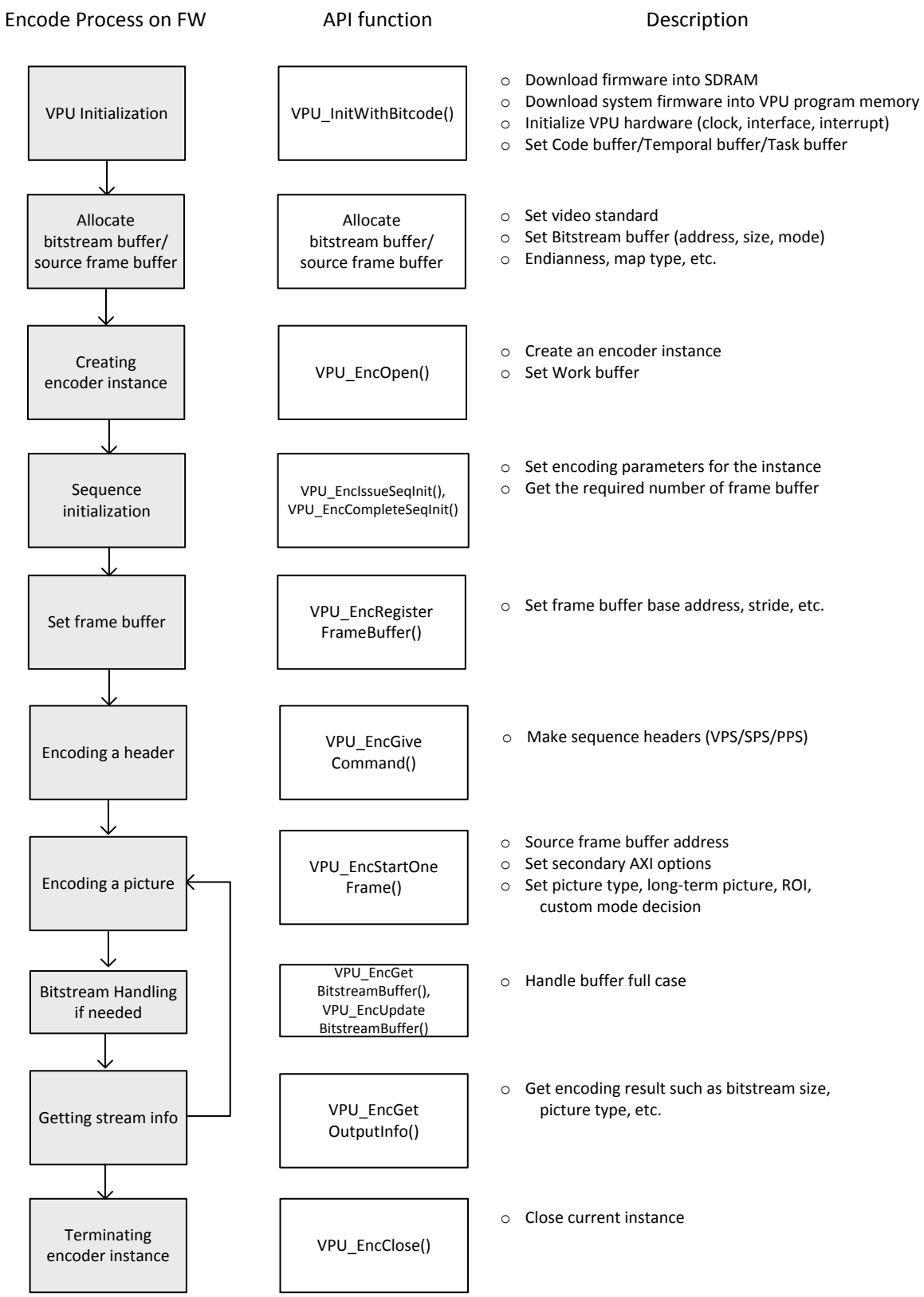


Figure 3.1. Encoder Control Flow with APIs

3.2.2. Creating an Encoder Instance

After initialization of VPU, the first step to run encoder operation is the creation of an encoder instance and the acquisition of the instance handle, so that VPU can identify which instance is now running in multiple instance environment. It could be easily done by using a single API function called `VPU_EncOpen()`.

When creating a new encoder instance, application should specify the internal features of this encoder instance through `EncOpenParam` structure. This structure includes the following information about the new encoder instance:

- Bitstream buffer address & size

  Physical start address of bitstream buffer and its size

- Codec standard

  A video codec standard such as H.265/HEVC

- Picture size

  A width and height of source picture to encode

- Target frame rate and target bitrate with initialDelay

- Dynamic buffer allocation enable

  Application can allocate picture bitstream buffer dynamically by enabling dynamic buffer allocation. The address and the size of picture bitstream buffer can be dynamically given by application when issuing picture encode command. For use of static bitstream buffer, the address and the size of bitstream buffer retrieved by `VPU_EncGetInitialInfo()` are used in picture encoding.

- Video standard specific parameters

  Specify standard-specific parameters for each video codec standard. For the details, please refer to the *API reference manual*. Most of the H.265/HEVC parameters for encoding are defined in EncHevcParam data structure.

- Profile, Level, and Tier

  VPU supports both main profile and main 10 profile. Host processor can add profile information to SPS by setting the profile register. However, if you set 0 or have done nothing to the register, VPU automatically encodes a profile by using the bit depth of source picture - main profile with 8bit and main10 profile with 10bit.

  VPU is able to support up to the level 5.0 with main and high tier. The level and tier information can also be given to SPS by setting the level and tier register repectively. When the level or tier register has 0, VPU can determine level and tier on its own and add a valid value to SPS.

- BitDepth and ChromaFormatIdc

  Host processor should give basic picture information such as pixel bitdepth and chroma format of source picture. Currently, VPU is able to take 8/10 bit 4:2:0 picture as its input.

- Lossless mode and constrained intra prediction mode

  VPU can encode in lossless mode for the application where no distortion is allowed in reconstructed frames. Also it supports constrained intra prediction.

- Cyclic GOP related parameters

VPU provides cyclic GOP parameters that are configurable for composing a wide range of coding structures. CyclicGopSize indicates the number of repeated picture(s) of coding structure possibly up to 8. VPU delivers CyclicGopPresetIdx for host processor to use commonly used coding structures (IPP, IBP, IBBP, etc).

- Temporal scalability

  VPU supports temporal scalability offering up to 7 temporal layers.

- Other various encoding tools

  VPU offers a variety of encoding tools like CuSizeMode, TmvpEnable, SkipIntraTrans, etc. so that application can use whatever fits their requirements.

  In addition, VPU has a suggested combination of encoding tools called a preset for easy configuration.

- Rate control parameters

  VPU is capable of rate control that takes bitrate, video quality, and buffer status into consideration at hierarchical levels - GOP, CTU, and Sub-CTU(32x32) level. For customizability, there are many parameters for rate control - TransRate, RcTargetRateLayer, CuLevelRcEnable, HvsQpEnable, HvsLambdaEnable, HvsQpScaleEnable, HvsQpScale, MinQp, MaxQp, and MaxDeltaQp.

  It also has other parameters such as HierachicalBitEnable and RcTargetRateLayer allowing accurate rate control based on temporal scalability using multiple layers.

- Slice enable/disable, slice mode and size

  VPU can encode a picture into bitstream with mulitple slices as many as configured. Also there can be slice segments within a slice.

  Host processor can define a slice with the number of CTB and a slice segment with the number of generated bits as well as CTB count. VPU generates multiple slices or slice segments unless the slice size register is 0.

- IntraRefresh

  IntraRefresh can be enabled for error robustness. Host application can specify the number of intra CTBs in a non-intra picture and IntraRefresh mode.

- ConformanceWindowOffsets

  VPU provides parameter for specifing the number of pixels to crop named ConformanceWindowOffsets (ConfWinLeft, ConfWinRight, ConfWinTop, ConfWinBot). It is for decoder to display only a rectangular region given by host processor.

By using these options, application can get the well-optimized encoding output that fits requirements of target application.

For example, in case that packet size is fixed, application might need to insert one slice to a certain amount of bits. However, it is hard for the application to make sure that output slice size is smaller than the given size because of variable length characteristics of encoding process.

For above reason, many applications should divide a slice into two packets frequently, which might cause big inefficiency in packetization. In order to achieve an easy packetization and effectiveness, application can set a slice size by (`packet_size` : N) with a certain margin of N, which allows output slice size to be less than the packet size. Then application can easily put a slice into a packet by just referring the slice boundary information provided by VPU as the encoder output.

After creating an encoder instance with these parameters, application cannot change these initial parameters specified in this stage. In principle, if application wants to change any of these basic parameters, it should close this instance and re-create another encoder instance with new initial parameters.

However, in practice, application needs to change some of these initial parameters depending on target application environment. So by providing a dynamic configuration command, VPU API enables application to configure part of these initial parameters dynamically. For the details, refer to the `VPU_EncGiveCommand()` description in the API reference manual.

In fact, the API function, `VPU_EncOpen()`, does not require any actual operation on VPU side but declares all the internal parameters to be used in later stage as well as bitstream buffer information.

3.2.3. Configuring VPU for Encoder Instance

3.2.3.1. Sequence Initialization

After registering all the required information for a new encoder instance, host application should configure VPU to be ready for supporting the new encoder instance. This procedure is done by setting all the encoder related information to Host Interface Registers of VPU and giving the command `ENC_SET_PARAM` to VPU for initiating internal configuration operation in VPU.

There are three modes of running ENC_SET_PARAM command - COMMON, GOP, and CHANGE_PARAM according to the ENC_SET_PARAM command option. For sequence initialization, host application should set ENC_SET_PARAM command(COMMON) parameter registers and give VPU the ENC_SET_PARAM command.

After then if host application wants to use a customized cyclic GOP instead of cyclic GOP preset, they should configure the relevant ENC_SET_PARAM command(GOP) parameter registers and give VPU the ENC_SET_PARAM command again.

All those processes are mainly done by the API function `VPU_EncGetInitialInfo()`. This function returns very crucial output parameters for encoder operation, the minimum number of frame buffers and the minimum number of source buffers.

Mostly this process does not require so much time, and it should be done only once at the beginning of encoder instance. It is not recommended to use interrupt signaling for this function, but interrupt signaling is allowed after completion of this operation by enabling the corresponding bit on Interrupt Enable Register.

3.2.3.2. Registering Frame Buffers

This configuring process is finished by registering frame buffers to VPU for future picture encoding operation. In this final stage of configuration, the returned parameter from `VPU_EncGetInitialInfo()`, the minimum number of frame buffer, has a very important meaning. This parameter means that application should reserve at least the same number of frame buffers for proper encoding operation - to save reconstructed frame. This configuration can be done by the API function `VPU_EncRegisterFrameBuffer()`.

3.2.3.3. Generating High-level Header Syntaxes

When opening an encoder instance is completed by calling `VPU_EncGetInitialInfo()`, application MUST generate the high-level header syntaxes such as VPS/SPS/PPS in HEVC from VPU by using `VPU_EncGiveCommand()`.

There are two possible ways for generating these header syntaxes. The recommended way for getting header syntaxes is to use `ENC_PUT_VIDEO_HEADER` command. If application uses this command, the header syntaxes as a result are stored into the bitstream buffer according to the given endian setting.

The other way is creation of header syntaxes while picture encoding is performed. 1 of implicitHeaderEncode allows to generate not only slice data but also header syntaxes when ENC_PIC command is executed. If nothing

changes ever since header syntaxes have been created, header syntaxes are not anymore be generated on later picture encoding.

3.2.4. Running Picture Encoder on VPU

3.2.4.1. YUV Input Loading

Before running a picture encoder operation, host application should provide 8bit or 10bit 4:2:0 input YUV images with pre-defined image size for H.265/HEVC.

The new input image might come from the external video input device like CMOS sensor. In order to prevent VPU from being idle while waiting for completion of receiving input source picture, it is recommended to use dual buffering scheme of input image. Then encoder does not spend any cycles for idling before starting encode operation.

3.2.4.2. Initiating Picture Encoding

When activating picture encoding operation, application should provide the following information to VPU:

- Source frame address

  A source frame address contains the base address of each component of input YUV picture, which includes luminance and chrominance (or Cb and Cr) frame buffer base address.

- Forced frame skip option

  Forced frame skip is to skip encoding a current frame unconditionally.

- Output information report

  Applications can enable or disable whether VPU generates the informative output data such as CU Info (CU QP, slice boundary) and Slice Info during encode operation.

- forcePicQpI, forcePicQpP, forcePicQpB

  Applications can force to set a quantization parameter for I, P, or B picture when rate control is disabled.

- Picture type

  A picture type (I, P, or B) is configurable for encoding a current picture. When use of cyclic GOP, the picture type pre-determined in cyclic GOP is ignored.

- IRAP picture (forceIPicture)

  A current picture can be set as IRAP picture (IDR or CRA) to work as random access point. This option changes a picture type of the current picture into an I picture.

Note It might not be available for the application where encoding order is different from display order (delay exists).

- Secondary AXI

  Set a secondary AXI option for bandwidth saving. There are secondary AXI options for encoding, which are useEncLfEnable and useEncRdoEnable

- Frame buffer

  Set a frame buffer mode such as CbCr interleave, NV21, and endianness.

- Implicit Header Encoding

  Set an implicit header encoding mode for generating bitstream conforming to HEVC standard. Receiving a new ENC_SET_PARAM command during encoding with change of the encoding tool associated parameters like disableDbk, VPU encodes and adds the new high level syntax (SPS or PPS) before bistream of the current picture.

After setting all these information to VPU, host processor can initiate picture encoding operation by sending ENC_PIC command to VPU. All of these pre-defined processes can be performed by just calling a single API function, `VPU_EncStartOneFrame()` with `EncParam` structure. This API function initiates picture encoding operation. Returning from this API does not mean that picture encoding is completed.

The quantization step size given to VPU with ENC_PIC_RUN is meaningful only when rate control option is disabled. This additional feature is provided to support application-specific VBR encoder operations.

The forced frame skip option is so useful when encoding a new picture is not allowed temporarily due to the external situation of encoder. That is to say, forced frame skip is used by application when encoding a picture is problematic under a certain external situation. For example, we can assume that channel condition is temporarily too bad and transmitting encoded stream is impossible. Then the application can suspend encoding operation for a while by using this Forced frame skip option.

3.2.4.3. Completion of Picture Encoding

Picture encoder operation takes a certain amount of time, and application could go on other tasks while waiting for the completion of picture encoding operation such as packetization of encoded stream for transmission. Application can use two different types of schemes for detecting completion of picture encoding operation: polling a status register and interrupt signaling.

When application is going to use a polling scheme, they just need to check the BusyFlag in the corresponding register. Calling `VPU_IsBusy()` gives the same result.

Interrupt signaling could be the most efficient way to check the completion of a given command. An interrupt signal for ENC_PIC command is mapped on bit[3] of Interrupt Enable Register. So application could easily know the completion of picture encoder operation from this dedicated interrupt signal from VPU.

3.2.4.4. Encoder Stream Handling

When an encoder bitstream buffer is big enough to store any big size of picture stream, then the encoder does not need to get any bitstream during picture encoder operation. After encoder operation is finished, host application can read the encoded bitstream according to the requirements of packetization.

But when an encoder bitstream buffer is not big enough to store a whole picture stream, encoder buffer-full can happen. Unless this buffer-full situation is resolved, the encoder task running on VPU might be hanged. So while picture encoding goes on, application should keep reading out encoded bitstream from the bitstream buffer in order to avoid this undesirable encoder hanging. Therefore when bistream buffer becomes full, VPU asserts an interrupt to inform host processor of this situation.

When using a limited size of encoder bitstream buffer, bitstream reading during encoder operation is recommended. By using two dedicated functions, `VPU_EncGetBitStreamBuffer()` and `VPU_EncUpdateBitStreamBuffer()`, application can easily handle the read pointer in encoder case while accessing the encoder bitstream buffer.

Note Host application needs to set at least 96 Kbytes of bitstream buffer for use of ring-buffer scheme due to VPU hardware constraint.

But if there is a big bitstream buffer (line buffer is used) enough to store one encoded picture data in order to avoid VPU from being hanged, then host processor could read encoded bitstream only after each picture

encoding has been completed. In this case, application could do safely other tasks while picture encoding is running on VPU. When use of this frame-based streaming option, `VPU_EncGetBitStreamBuffer()` and `VPU_EncUpdateBitStreamBuffer()` are useless.

3.2.4.5. Acquiring Encoder Results

Whenever a picture encoding is completed, host application can get the encoded output such as encoded bytes and encoded source index, recon frame index, etc. The API provides a function VPU_EncGetOutputInfo() for getting output results of picture encoder.

- Encoded source picture index

  With support of encoding into B pictures, there might be a gap between encoded order and display order - an encoded picture might not be from the source picture when ENC_PIC command is executed. For that reason, VPU manages a souce picture index and an encoded picture index separately as a frame buffer identifier and accesses or controls the frame buffers with each index.

  Also, source pictures are likely to wait to be encoded due to delay if B picture is used. VPU returns an encoded source picture index for the current ENC_PIC command so that host processor can recognize which source picture has been encoded this time.

  – If there was no encoded picture due to intial encoding delay, VPU returns -2 of source picture index.

  – If there is no more source picture, host processor should set 1 to srcEndFlag, by which VPU returns -1 of source picture index for ENC_PIC command.

- Reconstructed picture index

  By ENC_PIC command, VPU not only encodes a picture into bitstream, but also reconstructs the picture and saves it to the frame buffer for later use of refererence.

  Note VPU embeds FBC(Frame Buffer Compression) hardware block for bandwidth saving which compresses and stores the compressed format of recon picture to the frame buffer of external memory.)

  VPU returns a recon picture index indicating an index of the frame buffer where the reconstructed frame of current encoded picture is stored.

  – If there was no encoded picture due to intial encoding delay, VPU returns -2 of recon picture index.

  – If VPU returns -1 of recon picture index, it indicates end of encoding.

- Encoded cyclic GOP picture index

  Report on a picture index of cyclic GOP structure for the currently encoded picture when use of cyclic GOP

- Encoded picture POC

  A POC value of the currently encoded picture

- Encoded picture type

  A picture type of the currently encoded picture
  – 0 : I picture
  – 2 : P picture
  – 4 : B picture

- Encoded picture number

The number of pictures that have been encoded so far

- picture skip info

  Report on whether the current picture has been skipped or not

- Number of slice segments

  A picture can be composed of one or more slice segments. It reports how many slice segments are included in the currently encoded picture.

- Stream size of encoded picture in bytes

  Report on the nal unit byte size after completion of one picture encoding which is the total size of all nal units including VCL, SPS, PPS, etc.

- Number of Intra block

  The number of intra block (8x8 unit) within the currently encoded picture

- Number of skip block

  The number of skipped block (8x8 unit) within the currently encoded picture

- Average CTU QP

  The average QP value for all CTUs within the currently encoded picture

3.2.5. Terminating an Encoder Instance

When host application wants to finish encoder operation and terminate an encoder instance, they just can release the handle of the instance and terminate the instance by using the SEQ_END command. It can be done simply by calling `VPU_EncClose()` function.

3.2.6. Dynamic Configuration Commands

While running picture encoding operation sequentially, sometimes application needs to give a certain special commands to VPU such as change of parameter, insertion of high layer header syntaxes(VUI or HRD), etc. The VPU API provides a set of commands to support these kinds of special requests from host application.

Applications can change encoding options or high layer header syntaxes by calling VPU_EncGiveCommand() with ENC_SET_PARA_CHANGE.

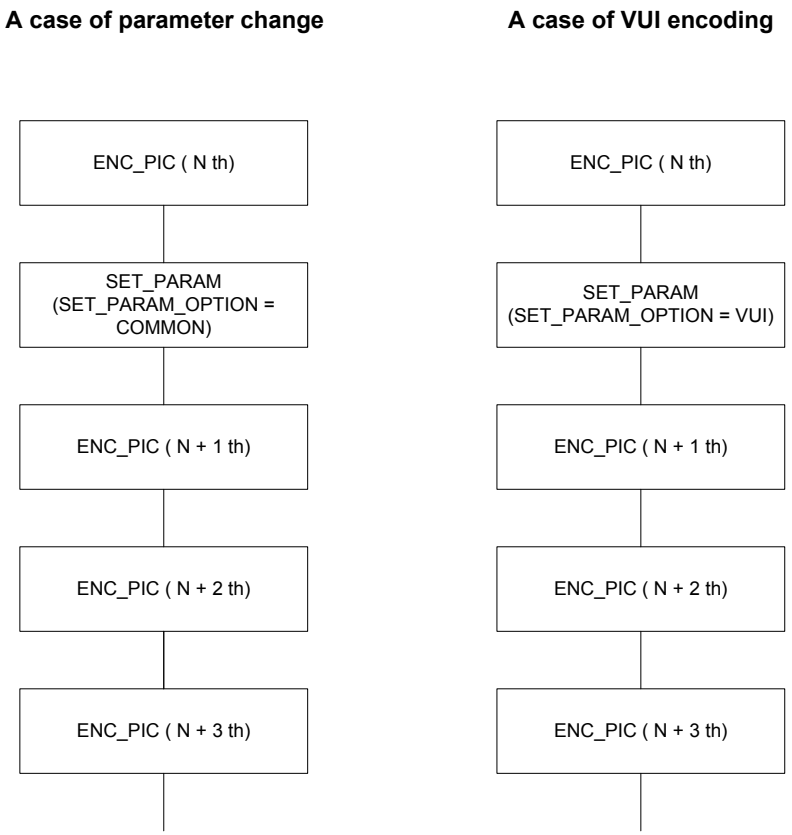


Figure 3.2. Change of Parameter While Encoding

The left figure of *Figure 3.2, "Change of Parameter While Encoding"* represents an example flow of encoding in which the command for adding a high level syntax (parameter change) is given in the middle of encoding.

After ENC_PIC #t command is issued, host processor gives VPU a new ENC_SET_PARAM command for changing encoding parameters. Then on ENC_PIC # t + 1,

- If implicit header encoding is 1, VPU decides high level syntax conforming to H.265/HEVC standard and inserts the SPS or PPS into stream.
- If implicit header encoding is 0, VPU reads high level syntax information from the code option register that is given by host processor and performs explicit header encoding.

Note In fact, ENC_SET_PARAM command only writes parameters to VPU internal memory, and actual header encoding with new parameters is done on next ENC_PIC command. If host processor wants to add a high level syntax without issuing a new ENC_SET_PARAM command, specify a header type (SPS or PPS or VPS) on code option register on ENC_PIC command.

Also the right figure of *Figure 3.2, "Change of Parameter While Encoding"* is an example of encoding with VUI. When host processor wants to put VUI information before or after encoded bitstream during encoding, VUI information should be written first on the relevant host interface register and the ENC_SET_PARAM command (w/ option = VUI) should be issued to VPU. Then on ENC_PIC # t + 1,

Appendix A
ERROR DEFINITION

A.1. System Error

The following indicates system errors that might occur while execution of command. VPU reports the system error information on RET_FAIL_REASON (BASE+0x10C) register when host processor sends QUERY(GET_VPU_INFO) command to VPU.

Table A.1. System Errors

Bit Pos	Bit Position	Error Reason	Originated Command	Description
0	0x0000_0001	ERR_QUEUEING_FAIL	SEQ_INIT, DEC_PIC/ ENC_PIC	The current command failed to be queued from fullness of command queue.
1	0x0000_0002	ERR_DECODER_FUSE		Reserved
2	0x0000_0004	ERR_INSTRUCTION_ACCESS_VIOLATION		Coprocessor0 exception
3	0x0000_0008	ERR_PRIVILEDGE_VIOLATION		Coprocessor0 exception
4	0x0000_0010	ERR_DATA_ADDR_ALIGNMENT		Coprocessor0 exception
5	0x0000_0020	ERR_DATA_ACCESS_VIOLATION		Coprocessor0 exception
6	0x0000_0040	ERR_GDI_ACCESS_VIOLATION	DEC_PIC/ ENC_PIC	Coprocessor0 exception
7	0x0000_0080	ERR_INSTRUCTION_ADDR_ALIGNMENT		Coprocessor0 exception
8	0x0000_0100	ERR_UNKNOWN		Unknown error
9	0x0000_0200	ERR_BUS_ERROR		Bus error
10	0x0000_0400	ERR_DOUBLE_FAULT		Double fault error
11	0x0000_0800	ERR_RESULT_NOT_READY	QUERY	The decode result cannot be retrieved, since picture decode operation has not been completed yet.
12	0x0000_1000	ERR_FLUSH_FAIL	FLUSH_INST	The FLUSH_INST command failed due to incompleteness of buffer flush.
13	0x0000_2000	ERR_UNKNOWN_CMD		Unknown command error

Bit Pos	Bit Position	Error Reason	Originated Command	Description
14	0x0000_4000	ERR_UNKNOWN_CODEC_STD		Unknown codec standard error
15	0x0000_8000	ERR_UNKNOWN_QUERY_OPTION	QUERY	The QUERY command failed with unknown query option.
16	0x0001_0000	ERR_VLC_BUF_OVERFLOW	DEC_PIC/ ENC_PIC	VLC buffer overflow occurred in parsing phase
17	0x0002_0000	ERR_TIMEOUT	DEC_PIC/ ENC_PIC	In case of H.265/ HEVC, it indicates timeout in parsing phase or in decoding phase. In case of VP9, it indicates timeout in decoding phase.
18	0x0004_0000	ERR_INVALID_TASK_NUM	DEC_PIC/ ENC_PIC	Invalid task buffer number error

Appendix B
POWER MANAGEMENT

For efficient power management, power-gating scheme is incorporated into the VPU(Video Processing Unit) design. This paper describes how to suspend and resume VPU in a seamless, safe way when it powers down or up by host processor's power management policy. It also covers shortly the VPU_SleepWake function in VPUAPI and implementation with OS.

B.1. Introduction

There are two terms regarding power saving techniques, power gating and clock gating that reader might get confused with. Power gating is that VPU is not completely provided with power, while clock gating only limits the clock from being given to certain modules of VPU.

When VPU is turned off, registers and internal memory in BPU pmem/dmem are removed. Then BIT processor stops working and gets into the state ready for restart. This is basically how VPU responds to power off.

The following chapters show a little details of how to control VPU safely at power down or up with some related codes.

B.2. At Power Down

1. Enable the VPU clock.

2. If VPU is running, wait until decoding or encoding operation is completely finished.

```
while (ReadVpuRegister(W5_VPU_BUSY_STATUS))
```

3. Send SLEEP_VPU command to let V-CPU flush the data of internal memory to external DRAM.

```
Wave5BitIssueCommand(core, W5_CMD_SLEEP_VPU);
```

4. Disable the VPU clock.

5. Power off the system.

B.3. At Power Up

1. Power up the system.

2. Enable the VPU clock source.

3. Reset VPU for returning to stable status since the power-up.

```
WriteVpuRegister(W5_VPU_RESET_REQ, 0x7ffffff) /* Reset All blocks */
```

4. Initialize the V-CPU Processor by issuing the INIT_VPU command so that VPU can be ready to get any command.

```
Wave5BitIssueCommand(core, W5_CMD_INIT_VPU);
```

5. Remap the code buffer.

6. Start the V-CPU Processor.

B.4. VPU_SleepWake() in VPUAPI

We offer VPU_SleepWake() function in the VPUAPI. But when target OS platform is Windows or Linux, device driver does directly handle this sort of function so that the VPU_SleepWake() function is not used. For implementation, host processor can refer to the linux driver code that we provide.

The VPU_SleepWake() function in VPUAPI can be used for the non OS platform or other OS platform where our application layer is able to handle a suspend/resume callback function.

B.5. Implementation with OS

On Linux or Windows OS platform, when VPU is suspended or resumed, power management module in kernel calls a callback function of device driver that has already been registered.

This is an example of implementation for power management in the Linux VPU device driver. It checks if VPU is running or not by polling the BIT_BUSY_FLAG register, instead of checking interrupt. Kernel scheduling API cannot be called in power management code.

Example B.1. Power Management Example Code in Linux Device Driver

```
vpu_suspend
{
    int i;
    int core;
    unsigned long timeout = jiffies + HZ;       /* vpu wait timeout to 1sec */
    int product_code;

    DPRINTK("[VPUDRV] vpu_suspend\n");

    vpu_clk_enable(s_vpu_clk);

    if (s_vpu_open_ref_count > 0) {
        for (core = 0; core < MAX_NUM_VPU_CORE; core++) {
            if (s_bit_firmware_info[core].size == 0)
                continue;
            product_code = ReadVpuRegister(VPU_PRODUCT_CODE_REGISTER);
            if (PRODUCT_CODE_W_SERIES(product_code)) {

                while (ReadVpuRegister(W5_VPU_BUSY_STATUS)) {
                    if (time_after(jiffies, timeout)) {
                        DPRINTK("SLEEP_VPU BUSY timeout");
                        goto DONE_SUSPEND;
                    }
                }

                Wave5BitIssueCommand(core, W5_CMD_SLEEP_VPU);

                while (ReadVpuRegister(W5_VPU_BUSY_STATUS)) {
                    if (time_after(jiffies, timeout)) {
                        DPRINTK("SLEEP_VPU BUSY timeout");
                        goto DONE_SUSPEND;
                    }
                }
                if (ReadVpuRegister(W5_RET_SUCCESS) == 0) {
                    DPRINTK("SLEEP_VPU failed [0x%x]", ReadVpuRegister(W5_RET_FAIL_REASON));
                    goto DONE_SUSPEND;
                }
            }
```

```
            else if (PRODUCT_CODE_NOT_W_SERIES(product_code)) {
                while (ReadVpuRegister(BIT_BUSY_FLAG)) {
                    if (time_after(jiffies, timeout))
                        goto DONE_SUSPEND;
                }

                for (i = 0; i < 64; i++)
                    s_vpu_reg_store[core][i] = ReadVpuRegister(BIT_BASE+(0x100+(i * 4)));
            }
            else {
                DPRINTK("[VPUDRV] Unknown product id : %08x\n", product_code);
                goto DONE_SUSPEND;
            }
        }
    }

    vpu_clk_disable(s_vpu_clk);
    return 0;

DONE_SUSPEND:

    vpu_clk_disable(s_vpu_clk);

    return -EAGAIN;
}


static int vpu_resume(struct platform_device *pdev)

{
    int i;
    int core;
    u32 val;
    unsigned long timeout = jiffies + HZ;       /* vpu wait timeout to 1sec */
    int product_code;

    unsigned long   code_base;
    u32                 code_size;
    u32                 remap_size;
    int                  regVal;
    u32                 hwOption  = 0;


    DPRINTK("[VPUDRV] vpu_resume\n");

    vpu_clk_enable(s_vpu_clk);

    for (core = 0; core < MAX_NUM_VPU_CORE; core++) {

        if (s_bit_firmware_info[core].size == 0) {
            continue;
        }

        product_code = ReadVpuRegister(VPU_PRODUCT_CODE_REGISTER);
        if (PRODUCT_CODE_W_SERIES(product_code)) {
            code_base = s_common_memory.phys_addr;
            /* ALIGN TO 4KB */
            code_size = (W5_MAX_CODE_BUF_SIZE&~0xfff);
            if (code_size < s_bit_firmware_info[core].size*2) {
                goto DONE_WAKEUP;
            }
```

```
        regVal = 0;
        WriteVpuRegister(W5_PO_CONF, regVal);

        /* Reset All blocks */
        regVal = 0x7ffffff;
        WriteVpuRegister(W5_VPU_RESET_REQ, regVal); /*Reset All blocks*/

        /* Waiting reset done */
        while (ReadVpuRegister(W5_VPU_RESET_STATUS)) {
            if (time_after(jiffies, timeout))
                goto DONE_WAKEUP;
        }

        WriteVpuRegister(W5_VPU_RESET_REQ, 0);

        /* remap page size */
        remap_size = (code_size >> 12) & 0x1ff;
        regVal = 0x80000000 | (W5_REMAP_CODE_INDEX<<12) | (0 << 16) | (1<<11) | remap_size;
        WriteVpuRegister(W5_VPU_REMAP_CTRL, regVal);
        WriteVpuRegister(W5_VPU_REMAP_VADDR,0x00000000);    /* DO NOT CHANGE! */
        WriteVpuRegister(W5_VPU_REMAP_PADDR,code_base);
        WriteVpuRegister(W5_ADDR_CODE_BASE, code_base);
        WriteVpuRegister(W5_CODE_SIZE,      code_size);
        WriteVpuRegister(W5_CODE_PARAM,             0);
        WriteVpuRegister(W5_INIT_VPU_TIME_OUT_CNT,   timeout);

        WriteVpuRegister(W5_HW_OPTION, hwOption);

        /* Interrupt */
        if (product_code == WAVE520_CODE) {
            regVal  = (1<<W5_INT_ENC_SET_PARAM);
            regVal |= (1<<W5_INT_ENC_PIC);
        }
        else {
            // decoder
            regVal  = (1<<W5_INT_INIT_SEQ);
            regVal |= (1<<W5_INT_DEC_PIC);
            regVal |= (1<<W5_INT_BSBUF_EMPTY);
        }

        WriteVpuRegister(W5_VPU_VINT_ENABLE,  regVal);

        Wave5BitIssueCommand(core, W5_CMD_INIT_VPU);
        WriteVpuRegister(W5_VPU_REMAP_CORE_START, 1);

        while (ReadVpuRegister(W5_VPU_BUSY_STATUS)) {
            if (time_after(jiffies, timeout))
                goto DONE_WAKEUP;
        }

        if (ReadVpuRegister(W5_RET_SUCCESS) == 0) {
            DPRINTK("WAKEUP_VPU failed [0x%x]", ReadVpuRegister(W5_RET_FAIL_REASON));
            goto DONE_WAKEUP;
        }
    }
    else if (PRODUCT_CODE_NOT_W_SERIES(product_code)) {

        WriteVpuRegister(BIT_CODE_RUN, 0);

        /*---- LOAD BOOT CODE*/
```

```
            for (i = 0; i < 512; i++) {
                val = s_bit_firmware_info[core].bit_code[i];
                WriteVpuRegister(BIT_CODE_DOWN, ((i << 16) | val));
            }

            for (i = 0 ; i < 64 ; i++)
                WriteVpuRegister(BIT_BASE+(0x100+(i * 4)), s_vpu_reg_store[core][i]);

            WriteVpuRegister(BIT_BUSY_FLAG, 1);
            WriteVpuRegister(BIT_CODE_RESET, 1);
            WriteVpuRegister(BIT_CODE_RUN, 1);

            while (ReadVpuRegister(BIT_BUSY_FLAG)) {
                if (time_after(jiffies, timeout))
                    goto DONE_WAKEUP;
            }
        }
        else {
            DPRINTK("[VPUDRV] Unknown product id : %08x\n", product_code);
            goto DONE_WAKEUP;
        }

    }

    if (s_vpu_open_ref_count == 0)
        vpu_clk_disable(s_vpu_clk);

DONE_WAKEUP:

    if (s_vpu_open_ref_count > 0)
        vpu_clk_enable(s_vpu_clk);

    return 0;
}
```

Appendix C
RATE CONTROL

WAVE encoder IP supports rate control which finds a quantization parameter (QP) adaptively to produce constant encoded bits. The rate control functions in three levels : picture level, CTU level, and subCTU level. The picture level RC determines a picture QP, the CTU level RC determines a 64x64 CTU QP, and the subCTU level RC determines a 32x32 block QP.

Besides normal rate control operation, WAVE encoder IP provides ROI (Region Of Interest) rate control option in the picture level.

C.1. Picture-level RC

Figure C.1, "Diagram of Picture-level Rate Control" represents how the rate control is performed in the picture level.

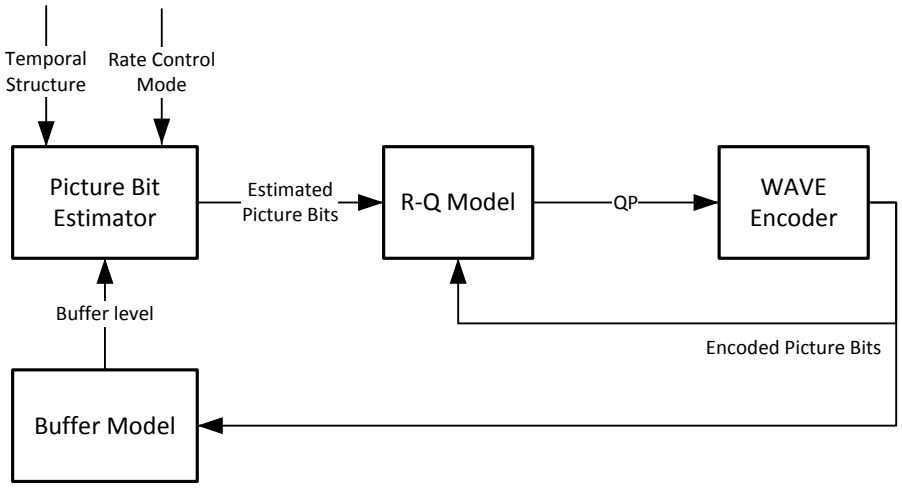


Figure C.1. Diagram of Picture-level Rate Control

A picture bit count is determined using buffer level, temporal structure, and rate control mode. R-Q model generates a QP with the determined picture bit count. WAVE encoder produces actually encoded picture bits which are fed both to R-Q model and to buffer model to refine the model parameters by comparing gap between estimated bits and encoded bits.

C.1.1. Buffer Model

The picture-level rate control is based on buffer model. The buffer model is controlled by three parameters: average bit rate, buffer size, and initial buffer level.

The buffer model size is determined by bitrate and InitialDelay which are sequence level parameters.

```
Buffer size = bitrate * InitialDelay
```

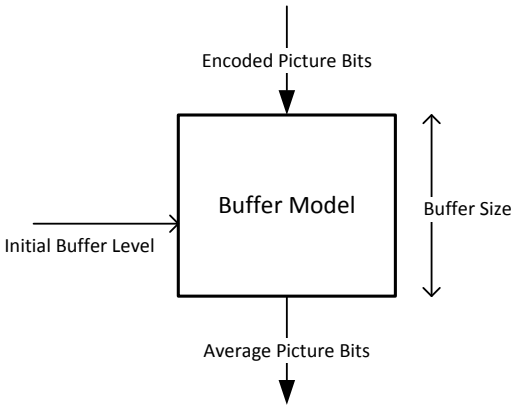


Figure C.2. Buffer Model

The buffer level is set to the initial buffer level at initial time. After encoding every picture, encoded bits are poured into the buffer, and average picture bits are drained from the buffer. The rate control controls QP so that the buffer does not suffer overflow or underflow.

C.1.2. Picture Bit Estimator

Picture Bit Estimator determines picture bits using buffer level, temporal structure, and rate control mode. When buffer level is around full, it decreases picture bits to prevent buffer overflow, while it increases picture bits when buffer level is around empty to prevent buffer underflow. By analyzing temporal structure, Picture Bit Estimator allocates more picture bits to more important or referenced pictures.

C.1.3. R-Q Model

R-Q model generates a picture QP by using R-Q equation inside. With the picture bit count from Picture Bit Estimator, the R-Q equation calculates a picture QP.

Also R-Q model gets feedback of actually encoded picture bits from Encoder and uses it to refine R-Q equation parameters for better QP estimation.

C.1.4. ROI

Host processor can define ROI (Region of Interest) with picture QP map. The QP map contains QPs for every 32x32 block. The allowed QP range is 0 to 51, inclusive. Specified regions of interest can have lower QPs than other regions in the QP map.

In rate control is enabled, WAVE5 encoder adjusts the QP values in the QP map proportionally to the RC estimated picture QP. *Figure C.3, "Adjustment of QP values by RC"* illustrates change of subCTU QP values after rate control is applied.

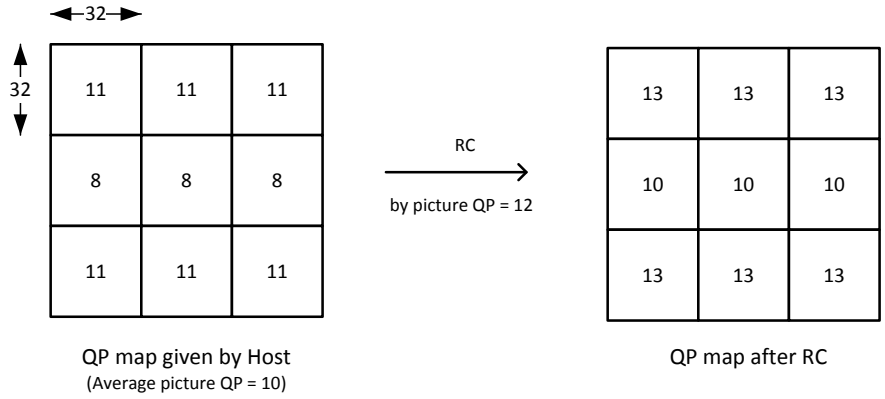


Figure C.3. Adjustment of QP values by RC

C.2. CTU-level RC

The CTU-level RC can regulate the buffer level more accurately than the picture-level RC does. CTU target bit is derived from the following formula. QP keeps changing for every 64x64 block when CTU-level rate control is enabled.

```
CTU_target_bit = picture_target_bit / num_CTU_in_picture
```

Figure C.4, "CTU-level RC with RC Buffer" describes how CTU-level rate control works with RC buffer.

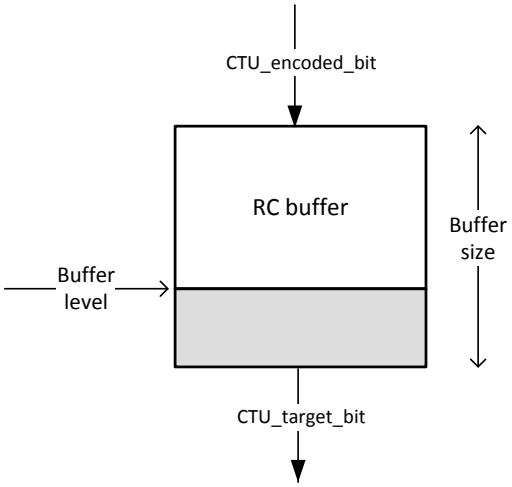


Figure C.4. CTU-level RC with RC Buffer

When CTU-level RC is enabled, CTU QP is decided based on the buffer level. CTU QP is linearly proportional to buffer level. In other words, CTU QP increases as buffer level goes up and CTU QP decreases as buffer level becomes low. The buffer level is controlled by accumulated encoded bit count of CTUs and complexity measure of CTUs.

When buffer level is full, CTU QP is assigned with max. QP. When buffer level is empty, CTU QP is assigned with min QP. Max QP and Min QP are sequence level parameters.

With small buffer size, i.e. small end-to-end delay, rate control with only picture level has possibility that does not meet the small delay constraint. CTU-level rate control is the solution to solve this problem. However, the CTU-level RC might bring undesirable video quality than picture-level RC due to the frequent QP change.

This function can be enabled by setting the EN_CU_LEVEL_RC register.

C.3. subCTU-Level RC

When subCTU-level rate control is used, QP is changed for every 32x32 block. It allocates a high QP value to complex block and a low QP value to static block, because human eyes are more sensitive to static area than complex area. Block complexity is measured by hardware block in the VCE which calculates variance of an original 32x32 block.

The subCTU-Level RC aims to improve subjective video quality while keeping a constant bitrate. So if this is used, SSIM score can be increased, but PSNR score might be decreased. Host processor can enable this by setting the relevant registers EN_HVS_QP and HVS_QP_SCALE.

C.4. Interface between Rate Control Levels

Figure C.5, "Interface between Rate Control Levels" describes each level of rate control and its interface on the architectural layer of WAVE encoder IP.

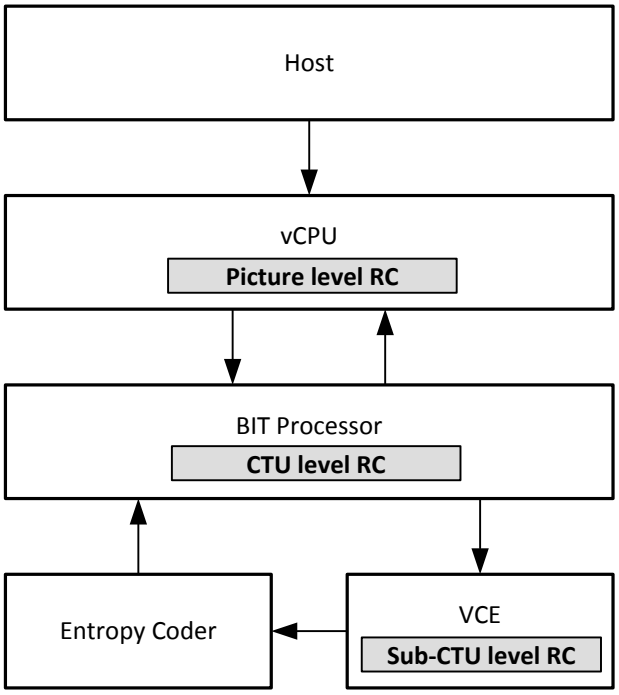


Figure C.5. Interface between Rate Control Levels

vCPU is a 32-bit processor that processes picture-level rate control. It is programmed using C code. BIT processor is a small programmable controller that processes CTU-level rate control. It is programmed using assembly code. VCE is the hardware unit that processes subCTU-level rate control and all pixel processing like motion estimation, transform, quantization and so on. Entropy coder is the hardware unit that processes syntax encoding.

C.5. Porting Customer's Rate Contol to WAVE Encoder

C.5.1. Porting Picture-level Rate Control

Figure C.6, "Porting Picture-level Rate Control" shows two ways of porting customer's picture-level rate control to the IP.

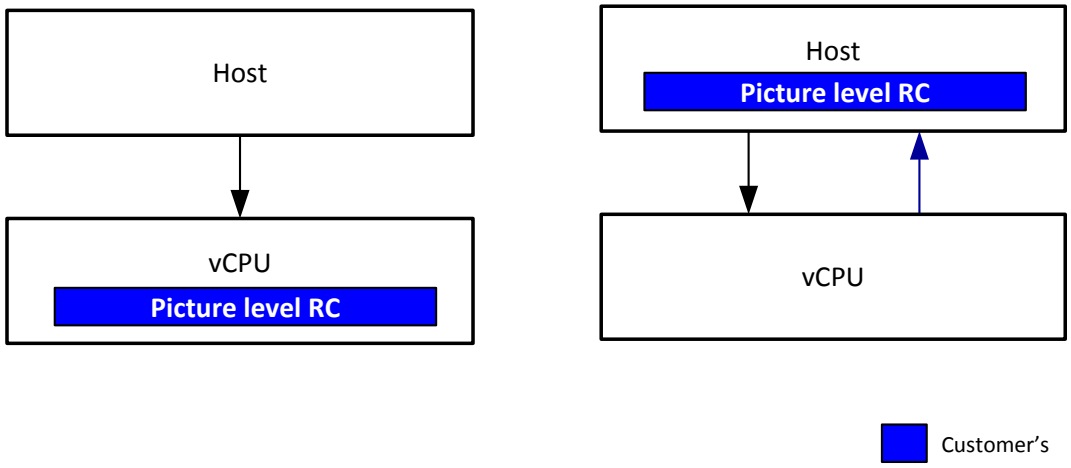


Figure C.6. Porting Picture-level Rate Control

- One is to implement the algorithm in vCPU. Customer delivers their algorithm and Chips&Media ports it to vCPU.
- The other is to implement the algorithm in host processor instead of vCPU. Host processor calculates picture QP using parameters from vCPU.

C.5.2. Porting CTU-level Rate Control

Host processor cannot be responsible for CTU-level rate control, because CTU-level interaction between host processor and the IP causes a lot of performance burden. *Figure C.7, "Porting CTU-level Rate Control"* depicts two feasible ways of porting customer's CTU-level rate control to the IP.

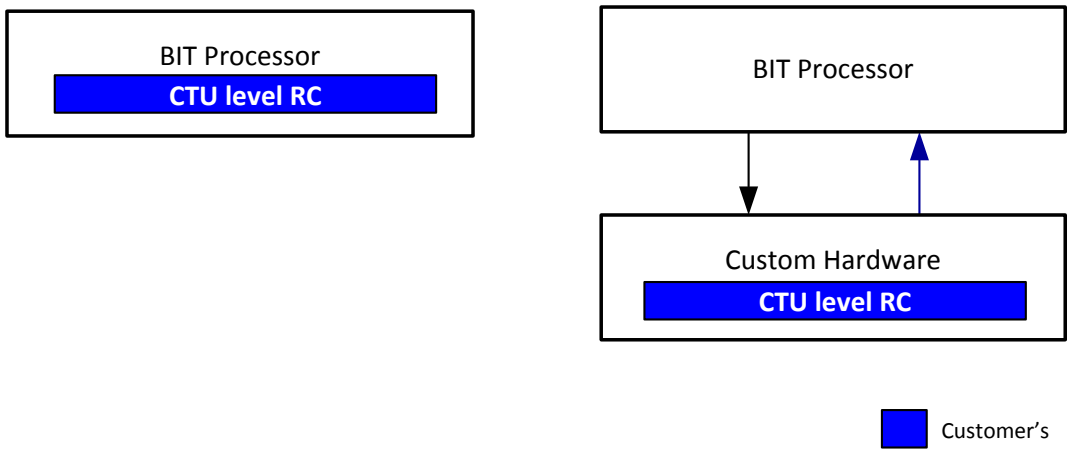


Figure C.7. Porting CTU-level Rate Control

Customer delivers their algorithm and Chips&Media ports it to BIT processor assembly code. If BIT processor is not fast enough to run the algorithm, custom hardware block for the algorithm might be necessary.

C.5.3. Porting subCTU-level Rate Control

There are two options for porting subCTU-level rate control to WAVE: designing custom hardware or implementing it in BIT processor. To implement it in BIT processor, the algorithm should be simple enough for BIT processor to run on time.

C.6. Quality Evaluation with C model

Chips&Media provides C model (.exe) that executes bit-exact operation as WAVE IP. With the C model, customer can simulate and evaluate their rate control algorithm before the algorithm is really ported to the IP.

There are two possible ways of allocating job and role between Chips&Media and customer for evalution of customer-specific RC algorithm on the WAVE C model.

- Customer explains their algorithm to Chips&Media. Then Chips&Media implements the algorithm and delivers an executable file to the customer.
- Chips&Media delivers the C code library for WAVE IP which excludes rate control. Then customer implements the C code for their rate control and builds an executable file by linking the C library with their C code. To work that way Chip&Media needs to get prototypes of rate control functions from customer.

Appendix D
SMART BACKGROUND ENCODING

In surveillance video, a lot of scenes are stationary or background. When video encoder compresses background detected area, it prefers skip mode or use of lower bits for the background.

It could be assumed that sophisticated video encoder automatically would assign skip mode to background. However, when there is noise in camera, this assumption might not work because noise prevents background from being encoded as skip mode.

To overcome such problem in video encoder side, usually ISP detects background and notifies it to video encoder. This scheme causes additional bandwidth and computational resources to be consumed.

Chips&Media's background encoding is implemented inside video encoder IP. With reuse of the existing encoder hardware block and on-the-fly processing, background detection does not require any additional bandwidth and computational resources.

D.1. BG Algorithm

Background encoding is composed of two modules, background detection and lambda adjustment.

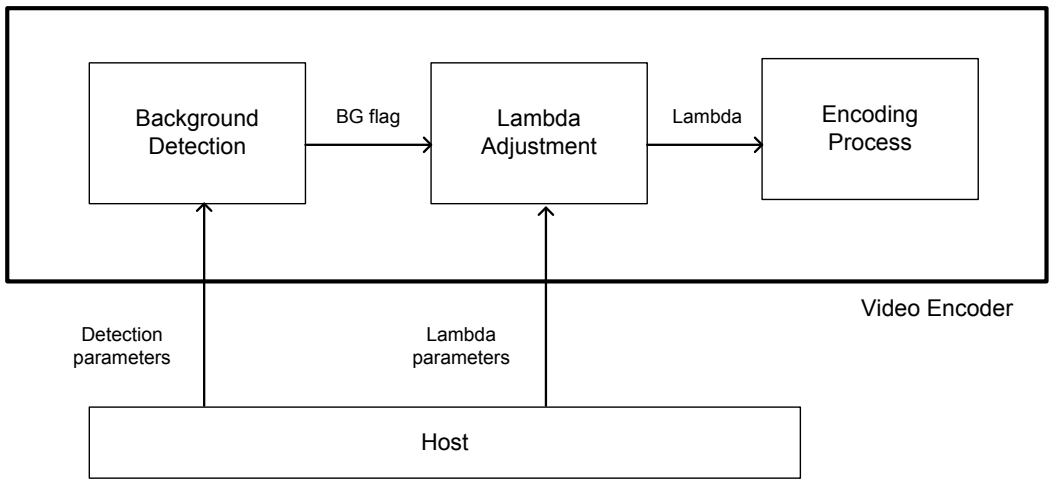


Figure D.1. Components of Background Encoding

The Background Detection module decides whether each 32x32 block is background or not and notifies BG flag to lambda calculation module. The BG flag is decided by calculating pixel differences between current frame and reference frame.

There are two control parameters that host processor can set for BG flag calculation, max pixel difference and average pixel difference. Host processor can set them freely, but recommended values are 8 for max pixel difference and 1 for average pixel difference.

The Lambda adjustment module calculates lambda for each 32x32 block. The lambda is an important parameter for mode decision in encoding process. When BG flag is enabled, lambda value is increased such that encoding process prefers skip mode to non-skip mode.

There are two control parameters that host processor can set, lambdaQP and deltaQP. The recommended values are 32 for lambdaQP and 3 for deltaQP. Lambda value to be delivered to encoding process is calculated as follows.

```
Lambda = QP_TO_LAMBDA_TABLE[max(lambdaQP, QP + deltaQP)]
```

where QP_TO_LAMBDA_TABLE is QP to lambda conversion table which is also used for non-background encoding. It is important that background encoding algorithm does not modify QP, but lambda. In the test we have carried out during algorithm development, lambda modification was shown better than QP modification in regards to image quality.

D.2. Relation with Rate Control

- **ROI**

  When ROI is enabled, background encoding is disabled.

- **CU level RC**

  There is no relation between CU-level RC and background encoding.

- **HVS_QP**

  HVS_QP assigns positive QP offset for complex area and negative QP offset for static area That is because human eyes are more sensitive to static area than complex area. With background encoding, HVS QP offset is adjusted such that background does not have negative HVS QP offset and non-background does not have positive HVS QP offset.

Note | Details of rate control are explained in *Appendix C, RATE CONTROL*.

D.3. BG Test Results

Figure D.2, "Detected Backgounds in Green" shows a still image of BridgeViewTraffic sequence. Green colored regions in the picture are backgrounds that have been detected by Chips&Media's BG module.

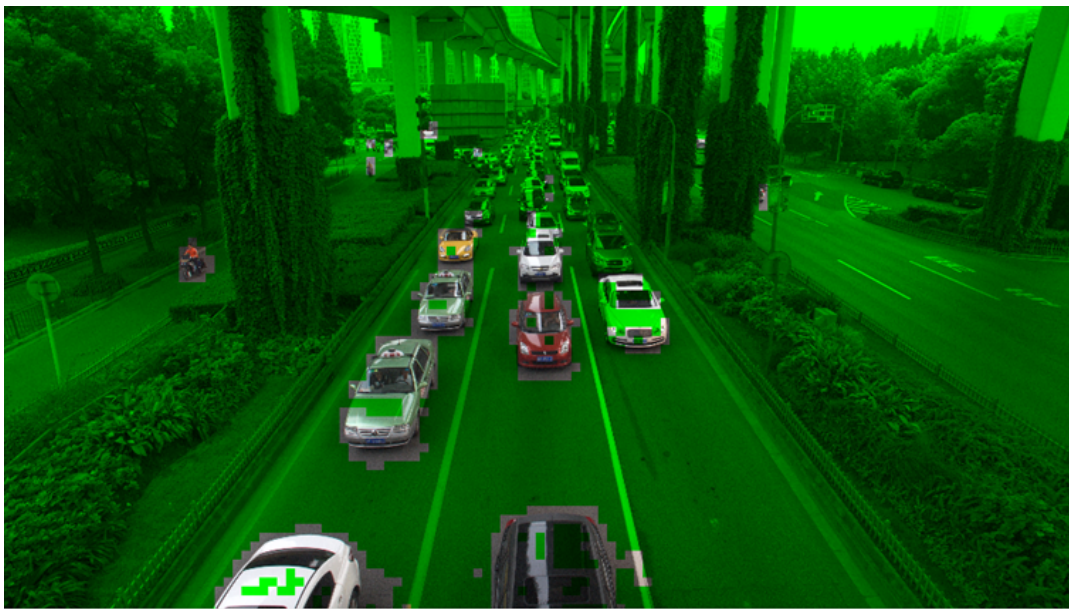


Figure D.2. Detected Backgounds in Green

Figure D.3, "Bitrate Saving from BG" shows the bitrate savings of three test sequences when background encoding is applied with constant qp of 22, 27, 32, and 37.

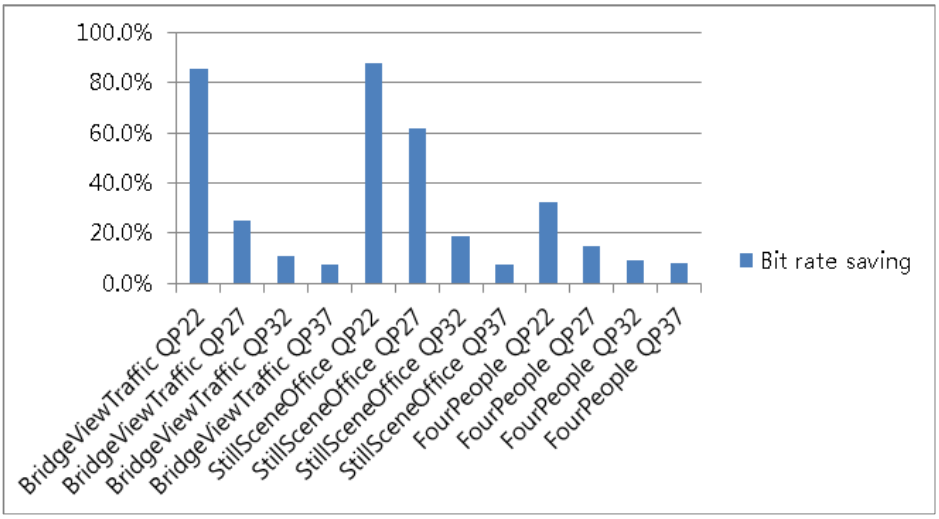


Figure D.3. Bitrate Saving from BG

Bitrate saving highly depends on bitrates and intra period. Bitrate saving is higher as bitrates are higher, because there are much bits to save in background at high bitrates. Also, bitrate saving is higher as intra period is longer, since background encoding is applied to only inter frame. In the test, intra period was set to 1 second.

D.4. Another coding tool for Background

Chips&Media HEVC encoder supports another tool for efficient encoding of background. That is long-term reference. Host processor can specify a good quality frame as long-term reference and use the frame as reference of the successive frames, which achieves improved image quality of background region. Long-term reference frame can be combined with background encoding for saving bit rates further.

Appendix E
CUSTOM MODE DECISION

WAVE5 HEVC encoding algorithm is designed for good video quality for a variety of applications. However, there are some applications that need customization of encoding algorithm to fit into their specific requirements. To meet this kind of demand, WAVE5 includes custom mode decisions as listed below.

- Custom lambda table
- Custom maps (QP, mode and lambda)
- Zero cost on/off
- Tuning mode decision

E.1. Custom Lambda Table

WAVE5 mode decision uses a well-known Lagrange multiplier method. It selects a mode such that

```
cost = distortion + lambda * rate
```

is minimized, where lambda value controls trade-off between distortion and rate. WAVE5 default lambda uses the following tables as a function of QP value.

```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 2, 2, 3, 4,
4, 6, 7, 9, 12, 15, 19, 24, 30, 38,
48, 61, 78, 99, 125, 159, 202, 256, 324, 412,
522, 663, 841, 1067, 1354, 1717, 2179, 2765, 3508, 4451,
5647, 7165
```

Figure E.1. Default Lambda

WAVE5 allows customers to set their own favorite lambda tables. It enables them to change relation between QP and bitrate.

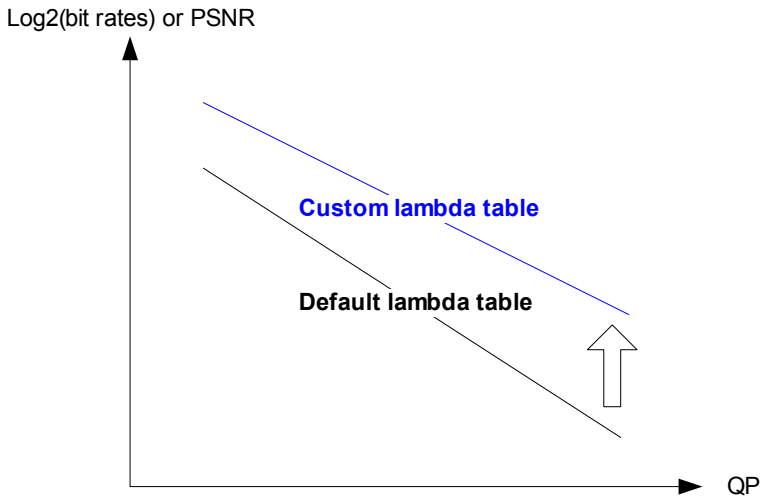


Figure E.2. Custom Lambda Table

E.2. Custom Maps

Custom maps can be delivered to encoder before encoding every frame. It modifies the default behavior of mode decision in encoder. There are three custom maps: QP map, mode map and lambda map.

E.2.1. QP Map

The QP map contains QPs for every 32x32 block. The allowed QP range is 0 to 51, inclusive.

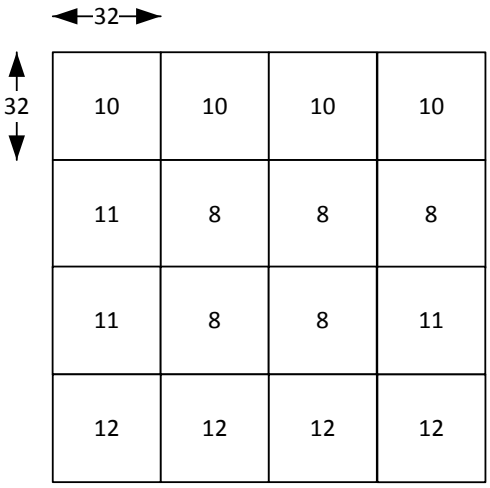


Figure E.3. Custom QP Map

E.2.2. Mode Map

The Mode map contains CU modes for every 64x64 block. The allowed values are 0(not use), 1(skip mode) and 2(intra mode).

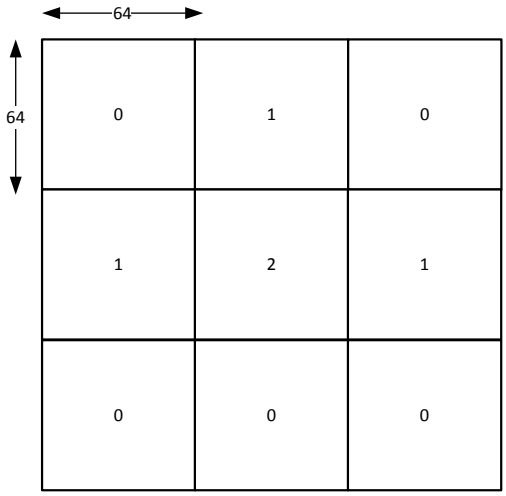


Figure E.4. Custom Mode Map

E.2.3. Lambda Map

The Lambda map contains lambda for every 32x32 block. The allowed values are 0 to 127, inclusive.

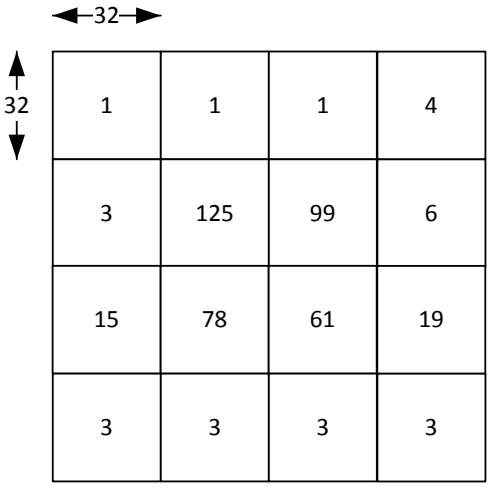


Figure E.5. Custom Lambda Map

E.3. Zero Cost On/Off

The mode decision of WAVE5 includes a special function that removes or sets all transform coefficients to zero and checks if the RD cost is better than ones in other encoding modes. By zeroing all transform coefficients, bitrate can be saved with sacrificing image quality. Since these saved bits are transferred to other blocks, average image quality can be improved.

However, in some application this function might not be good in terms of subjective quality, because image quality of some zeroing blocks can be unpleasant to human eyes. Another drawback of this function is that it reduces the acceptable QP range as indicated in *Figure E.6, "Zero Cost On/Off"*. WAVE5 has an option for this function to be on or off.

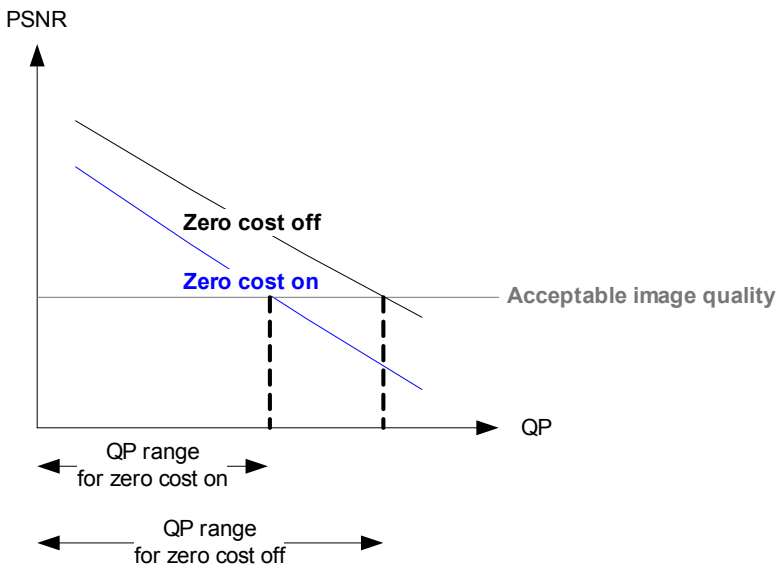


Figure E.6. Zero Cost On/Off

E.4. Tuning Mode Decision

WAVE5 mode decision basically uses the following formula as cost function.

```
RD cost = distortion + lambda * rate
```

Also, there is a possible way of tuning this cost function with delta rates. Customer can give each delta rate that affects the RD cost in intra mode decision, CU mode decision, and PU size decision.

```
RD cost = distortion + lambda * (rate + delta rate).
```

Table E.1. Delta Rates for Tuning Mode Decision

Category	Name	Range	Description
Intra modes	PU04IntraPlanarDeltaRate	0 ~ 255	Delta rates for planar mode of PU 4x4
	PU04IntraDcDeltaRate	0 ~ 255	Delta rates for DC mode of PU 4x4
	PU04IntraAngleDeltaRate	0 ~ 255	Delta rates for angular mode of PU 4x4
	PU08IntraPlanarDeltaRate	0 ~ 255	Delta rates for planar mode of PU 8x8
	PU08IntraDcDeltaRate	0 ~ 255	Delta rates for DC mode of PU 8x8
	PU08IntraAngleDeltaRate	0 ~ 255	Delta rates for angular mode of PU 8x8
	PU16IntraPlanarDeltaRate	0 ~ 255	Delta rates for planar mode of PU 16x16
	PU16IntraDcDeltaRate	0 ~ 255	Delta rates for DC mode of PU 16x16
	PU16IntraAngleDeltaRate	0 ~ 255	Delta rates for angular mode of PU 16x16
	PU32IntraPlanarDeltaRate	0 ~ 255	Delta rates for planar mode of PU 32x32
	PU32IntraDcDeltaRate	0 ~ 255	Delta rates for DC mode of PU 32x32
	PU32IntraAngleDeltaRate	0 ~ 255	Delta rates for angular mode of PU 32x32
CU modes	CU08IntraDeltaRate	0 ~ 255	Delta rates for intra mode of CU 8x8
	CU08InterDeltaRate	0 ~ 255	Delta rates for inter mode of CU 8x8
	CU08MergeDeltaRate	0 ~ 255	Delta rates for merge mode of CU 8x8

Category	Name	Range	Description
	CU16IntraDeltaRate	0 ~ 255	Delta rates for intra mode of CU 16x16
	CU16InterDeltaRate	0 ~ 255	Delta rates for inter mode of CU 16x16
	CU16MergeDeltaRate	0 ~ 255	Delta rates for merge mode of CU 16x16
	CU32IntraDeltaRate	0 ~ 255	Delta rates for intra mode of CU 32x32
	CU32InterDeltaRate	0 ~ 255	Delta rates for inter mode of CU 32x32
	CU32MergeDeltaRate	0 ~ 255	Delta rates for merge mode of CU 32x32
PU sizes	PU04DeltaRate	0 ~ 255	Delta rates for PU 4x4
	PU08DeltaRate	0 ~ 255	Delta rates for PU 8x8
	PU16DeltaRate	0 ~ 255	Delta rates for PU 16x16
	PU32DeltaRate	0 ~ 255	Delta rates for PU 32x32

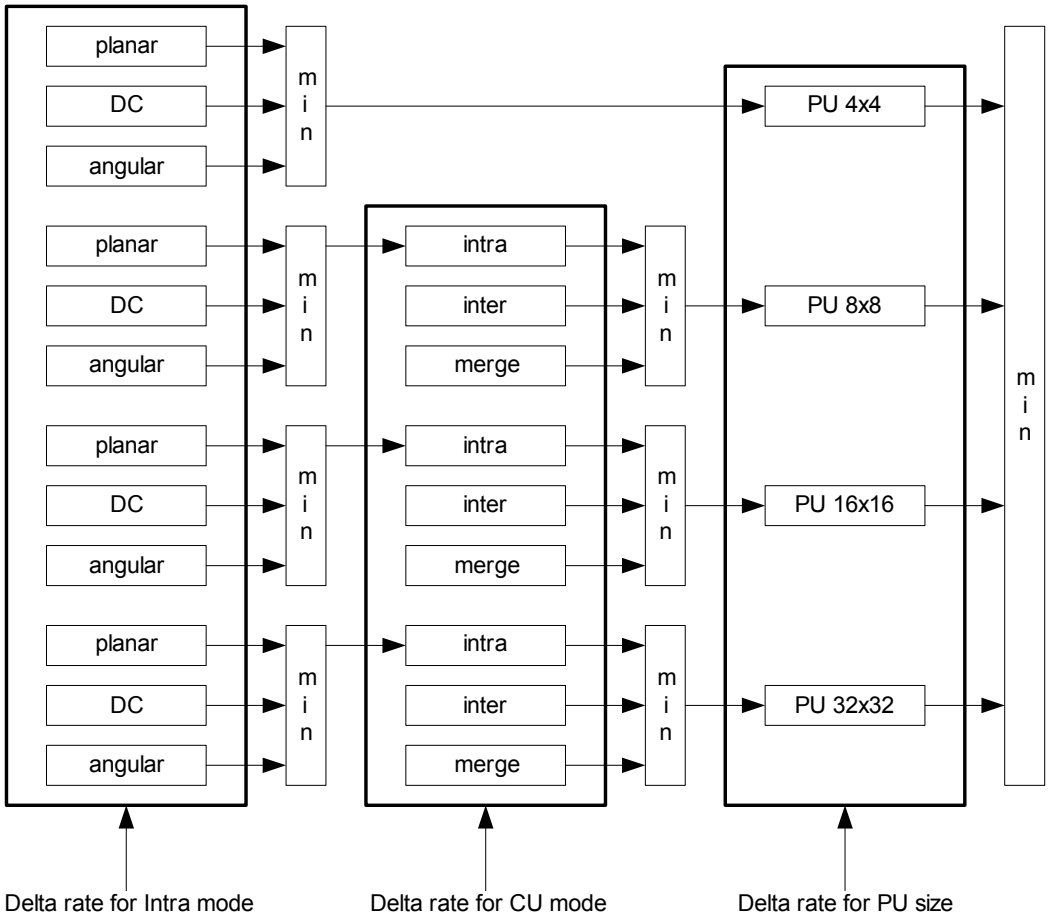


Figure E.7. Tuning Mode Decision

Example 1

In order to reduce intra blocks, customer can set like below.

- CU08IntraDeltaRate = 1
- CU16IntraDeltaRate = 4
- CU32IntraDeltaRate = 16.

Example 2

Customer can set 1 to PU04DeltaRate to reduce PU 4x4.

Appendix F
CHANGE OF ENCODE PARAMETER

Host processor can dymically change encoding parameters during encoding such as change of target bitrate on network.

There are some parameters that are allowed to be changed in the same sequence, but some could not be changed including source picture size, SPS and GOP parameters. If Host processor wants to change any of those parameters, it is dealt with sequence change inside VPU. If sequence change happens, it requires VPU to execute from CREATE_INST command and the following SET_FB, SET_PARAM and ENC_PIC command for the new sequence.

In order to change within the same sequence, Host processor should send ENC_SET_PARAM command with 0x10 of SET_PARAM_OPTION (CHANGE_PARAM). Then, VPU L1 firmware recognizes that "CHANGE_PARAM" command has taken place and starts encoding pictures with the new parameters as soon as getting "Change Param" command.

F.1. Changable Parameters

Table F.1, "CMD_ENC_SET_PARAM_ENABLE (0x118)" is a change enable flag register to set which parameter change is to be applied. Host processor can enable any of bit-field flag in this register; multiple selections also work. For example, for change of bitrate while encoding, Host processor should set ENABLE_SET_RC_TARGET_RATE and give a new bitrate to CMD_ENC_RC_TARGET_RATE register.

It is divided into three categories : RDO paramter registers, rate control parameter registers and PPS parameter registers.

Table F.1. CMD_ENC_SET_PARAM_ENABLE (0x118)

Bit position	Bit-field name	Related registers	ChangeParam type
[22]	ENABLE_SET_CUSTOM_LAMBDA	CMD_ENC_CUSTOM_LAMBDA_ADDR	ChangeRdoParam
[21]	ENABLE_SET_CUSTOM_MD	CMD_ENC_CUSTOM_MD_PU04 CMD_ENC_CUSTOM_MD_PU08 CMD_ENC_CUSTOM_MD_PU16 CMD_ENC_CUSTOM_MD_PU32 CMD_ENC_CUSTOM_MD_CU08 CMD_ENC_CUSTOM_MD_CU16 CMD_ENC_CUSTOM_MD_CU32	
[20]	ENABLE_SET_BG_PARAM	CMD_ENC_BG_PARAM	
[19]	ENABLE_SET_NR_PARAM	CMD_ENC_NR_PARAM CMD_ENC_NR_WEIGHT	
[18]	ENABLE_SET_SEQ_RDO_PARAM	CMD_ENC_SEQ_RDO_PARAM	
[17]	ENABLE_SET_SEQ_DEPENDENT_ SLICE	CMD_ENC_SEQ_DEPENDENT_SLICE	
[16]	ENABLE_SET_SEQ_INDEPENDENT_ SLICE	CMD_ENC_SEQ_INDEPENDENT_SLICE	
[11]	ENABLE_SET_RC_BIT_RATIO_ LAYER	CMD_ENC_RC_BIT_RATIO_LAYER_0_3 CMD_ENC_RC_BIT_RATIO_LAYER_4_7	ChangeRcParam

Bit position	Bit-field name	Related registers	ChangeParam type
[10]	ENABLE_SET_RC_MIN_MAX_QP	CMD_ENC_RC_MIN_MAX_QP CMD_ENC_RC_INTER_MIN_MAX_QP	
[9]	ENABLE_SET_RC_PARAM	CMD_ENC_RC_PARAM	
[8]	ENABLE_SET_RC_TARGET_RATE	CMD_ENC_RC_TARGET_RATE	
[0]	ENABLE_SET_PPS_PARAM	CMD_ENC_SEQ_PPS_PARAM	ChangePpsParam

F.1.1. Changable PPS Parameters

Table F.2. CMD_ENC_SEQ_PPS_PARAM

Bit position	Bit-field name	Description	Change-able
[31:10]	TC_OFFSET_DIV2	TcOffsetDiv2 for deblocking filter	O
[9:6]	BETA_OFFSET_DIV2	BetaOffsetDiv2 for deblocking filter	O
[5]	DISABLE_DBK	Disables the in-loop deblocking filter.	O
[4]	USE_WPP	Enables wave-front parallel processing.	X
[3]	USE_WP	Use weighted prediction.	O
[2]	USE_LF_CROSS_SLICE_ BOUND-ARY	Enables the use of in-loop filtering across slice boundaries	O
[1]	CONSTRAINED_INTRA_PRED	Enables constrained intra prediction.	O
[0]	USE_LOSSLESS_CODING	Enables lossless coding.	X

F.1.2. Changable RC Parameters

Table F.3. CMD_ENC_RC_PARAM

Bit position	Bit-field name	Description	Change-able
[31:20]	INITIAL_DELAY	Specifies an inital CPB delay in msec.	O
[19:14]	RC_INITIAL_QP	Sets an initial QP. If this value is smaller than 0 or larger than 51, the initial QP is decided by F/W.)	X
[13]	EN_SEQ_ROI	Enables ROI in sequence level.	X
[8]	BIT_ALLOC_MODE	Specifies picture bits allocation mode.	X
[7:4]	HVS_QP_SCALE	QP scaling factor for CU QP adjustment with 1 of en_hvs_qp.	O
[3]	EN_HVS_QP_SCALE	Enables QP scaling factor for CU QP adjustment with 1 of en_hvs_qp = 1.	O
[2]	EN_HVS_QP	Enables CU QP adjustment for subjective quality en-hancement.	O
[1]	EN_CU_LEVEL_RC	Enables CU level rate control.	X
[0]	EN_RATE_CONTROL	Enables rate control.	X

Other than CMD_ENC_RC_PARAM, there are other rate control parameter registers. Host processor can change any bit-field of the rate control registers during encoding.

- CMD_ENC_RC_MIN_MAX_QP

- CMD_ENC_RC_BIT_RATIO_LAYER_0_3
- CMD_ENC_RC_BIT_RATIO_LAYER_4_7
- CMD_ENC_RC_INTER_MIN_MAX_QP

F.1.3. Changable RDO Parameters

Table F.4. CMD_ENC_SEQ_RDO_PARAM

Bit position	Bit-field name	Description	Change-able
[22]	EN_MONOCHROM	Enables monochrom encoding mode.	X
[21]	USE_CUSTOM_LAMBDA	Enables custom lambda table.	O
[20]	USE_CUSTOM_MD	Enables custom mode decision.	O
[19:18]	MAX_NUM_MERGE	Sets maximum number of merge candidates. (0 ~ 2)	O
[8]	USE_INTRA_NXN	Enables intra nxn.	O
[7:5]	USE_CU_SIZE	Enables CU size. [0] : 8x8 CU [1] : 16x16 CU [2] : 32x32 CU	X
[4]	DISABLE_COEF_CLEAR	Disables the transform coefficient clearing algorithm in P or B picture. 0 : All-zero coefficient block is evaluated additionally in RDO 1 : All-zero coefficient block is not evaluated in RDO	O
[1:0]	USE_RECOMMEND_ENC_PARAM	Uses preset encoding mode in which tuned parameters are defined and used for each mode. 0 : custom 1 : recommend enc params (highest quality, slow encoding speed) 2 : boost mode(normal quality, normal speed) 3 : fast mode (low quality, fast encoding speed)	X

These are the paramter registers which belongs to ChangeRdoParam category. All they can be changed during encoding, except CMD_ENC_BG_PARAM[0] which enables background detection.

- CMD_ENC_SEQ_DEPENDENT_SLICE
- CMD_ENC_SEQ_INDEPENDENT_SLICE
- CMD_ENC_CUSTOM_MD_XXXX
- CMD_ENC_NR_PARAM
- CMD_ENC_NR_WEIGHT
- CMD_ENC_BG_PARAM
- CMD_ENC_CUSTOM_LAMBDA_ADDR

F.2. Restrictions

Changes go into effect in GOP unit

There is a restriction that change of parameter only goes effective in GOP unit. In other words, a new parameter is applied from the anchor picture of GOP.

Without such restriction, other troublesome or unecessary control might be needed when encoding non-low delay GOP structure (decoding order and display order are not same). For instance, if host processor issues ChangePpsParam command for every picture, VPU needs to encode PPS multiple times or to give a unique PPS ID as picture parameter changes.

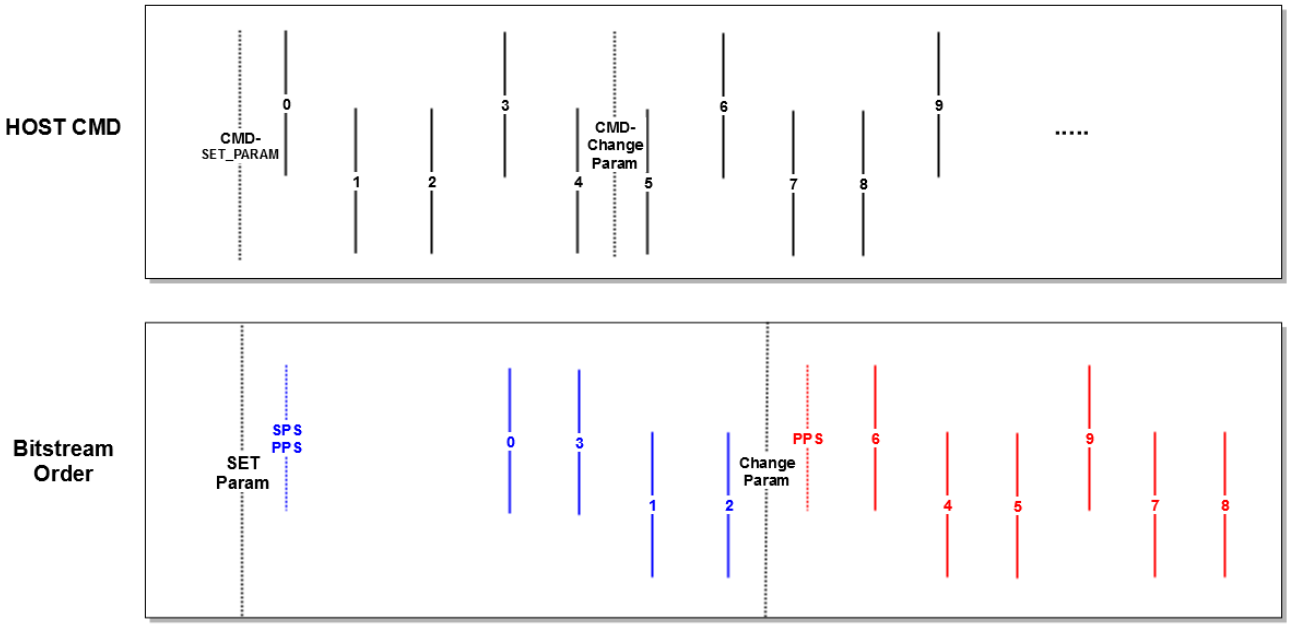


Figure F.1. Application of Parameter Change

Applying multiple changes

- In case that multiple changes on the same parameter happened in the same GOP, VPU encodes only with the final change. See *Figure F.2, "Multiple Changes on Parameters"*.
- In case that multiple changes on the different parameters take place, all the changes are gathered inside and applied from encoding anchor at once.

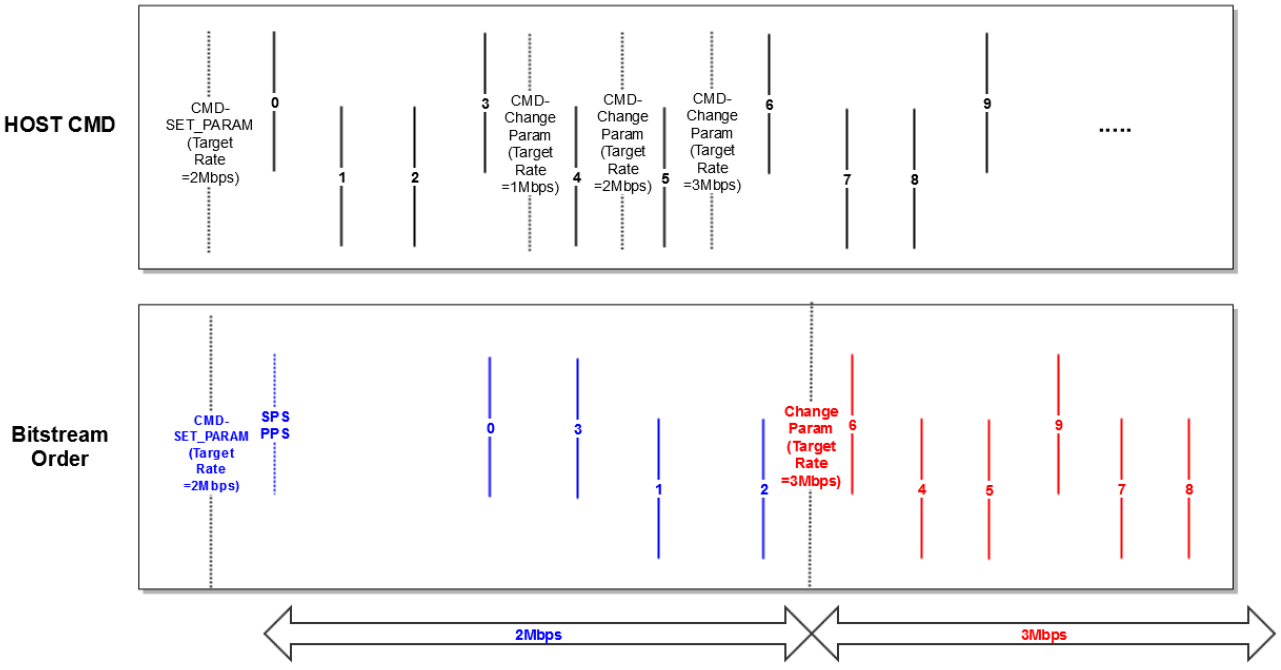


Figure F.2. Multiple Changes on Parameters

About Chips&Media

Chips&Media, Inc. is a leading video IP provider. Over the past decade, they have been doing high quality product design focusing on hardware implementation of high speed, low power, cost effective video solution for H.264, MPEG-4, H.263, MPEG-1/2, VC-1, AVS, MVC, VP8, Theora, Sorenson, and up to recent H.265/HEVC along with its remarkable bandwidth reduction technology.

They have a broad range of IP products from low-end 1080p HD mobile solutions to high-end 4K 60fps HEVC solutions or even brandnew HEVC/H.265, VP9, and AVS2 multi-decoder IP to fulfill target demands from various multimedia device markets.

Chips&Media's IPs are field-proven in over a hundred million of multimedia devices and by more than 60 top-tier semiconductor companies. The headquarter is located in Seoul, Korea (Republic of) with sales offices in Japan, China, and Taiwan. To find further information, please visit the company's web site at *http://www.chipsnmedia.com*