BITMAIN

Chips&Media™

WAVE510 HEVC Decoder IP

Programmer's Guide

Version 0.9.6

WAVE510 HEVC Decoder IP: Programmer's Guide

Version 0.9.6

Revision History

Date	Revision	Change
2016/8/8	0.9.0	Draft version generated
2017/3/17	0.9.1	Appendix. VP9 Superframe was included
2017/7/24	0.9.2	HOST INTERFACE chapter was updated.
2017/8/9	0.9.3	Appendix J: ERROR DEFINITION was updated.
2017/8/22	0.9.4	Figure 3.1. Decoder Control Flow with APIs was updated.
2017/8/24	0.9.5	CMD_DEC_SEI_MASK fields were added.
2017/9/15	0.9.6	CMD_DEC_USE_SEC_AXI register address changed to 0x150.

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 WAVE510 Programmer's Guide and its reference documentation. It contains the following sections:

1. About This Document

This document is the programmer's guide for WAVE510 HEVC Decoder 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

- *WAVE510 Datasheet*
- *WAVE510 Verification Guide*
- *WAVE510 API Reference Manual*

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.

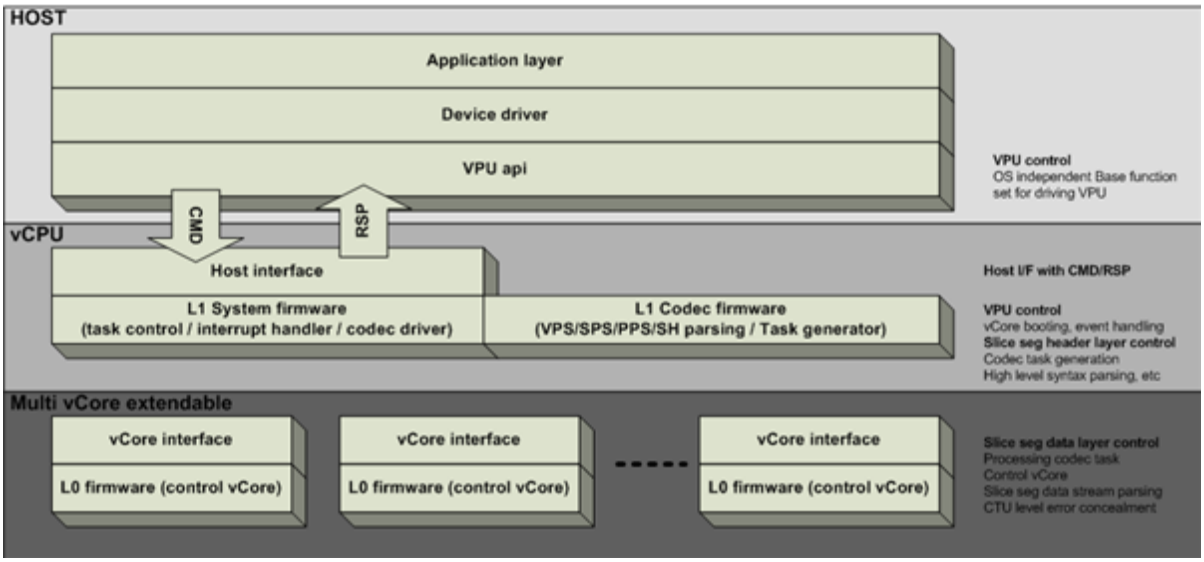


Figure 1.1. Exchanging Command/Response between Host and VPU

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 is a restriction that the base buffer addresses should be aligned in a 4KB unit, and 0xFFFF0000 to 0xFFFFFFFF (based on 32-bit) should not be assigned 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 1.2.3.3, "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 1.2.4.2, "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.

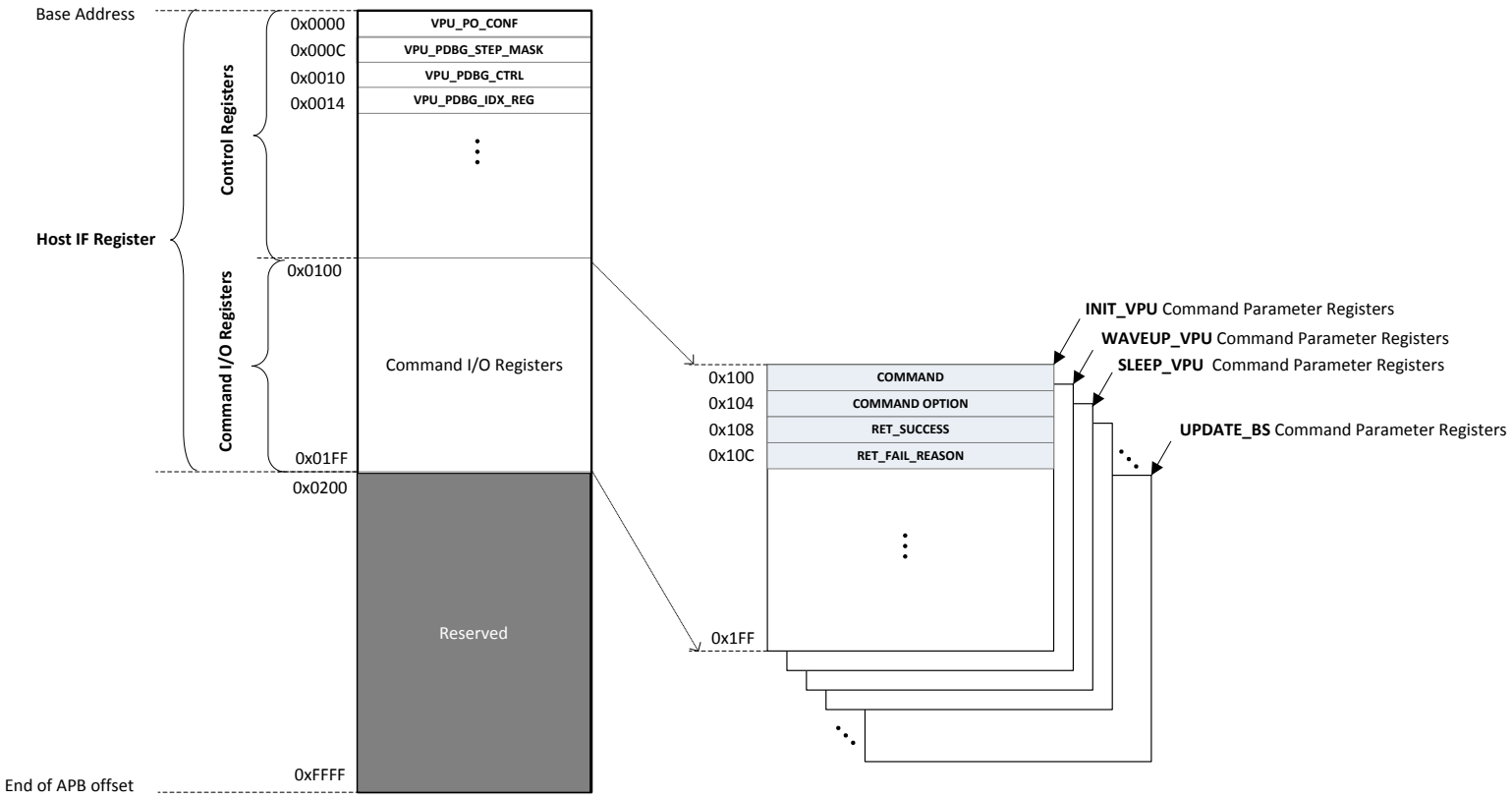


Figure 1.2. Host Interface Memory Map

1.2.2. Initialization Process

TBD

1.2.3. 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.

1.2.3.1. 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.

1.2.3.2. 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.

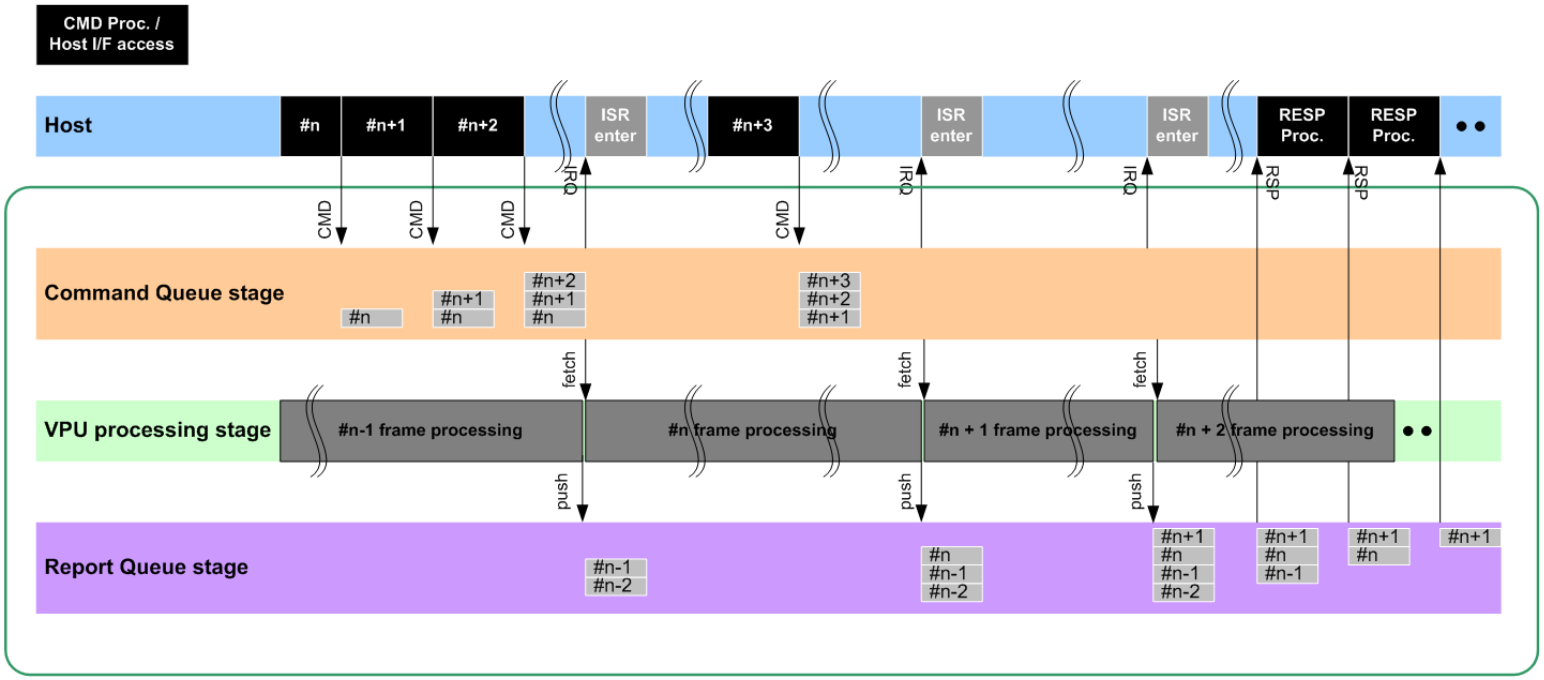


Figure 1.3. VPU's Internal Elastic Pipelines in Command Queue Architecture

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.

1.2.3.3. 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 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.

1.2.3.3.1. 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.

1.2.3.3.2. 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.

1.2.3.3.3. 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.

1.2.3.3.4. 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 (0x038) 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.

1.2.4. Summary of Host Interface Registers

1.2.4.1. 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
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	*VPU_PRESCAN_STATUS*	VPU_PRESCAN_STATUS
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
0x000000A8	RW	0x0	*RET_CONF_REVISION*	The revision of H/W configuration

Offset	Type	Reset Value	Name	Description
0x000000AC	RW	0x0	*RET_CONF_TYPE*	The define value of H/W configuration
0x000000F0	RW	0x0	*VPU_DBG_REG0*	Debug register #0
0x000000F4	RW	0x0	*VPU_DBG_REG1*	Debug register #1
0x000000F8	RW	0x0	*VPU_DBG_REG2*	Debug register #2
0x000000FC	RW	0x0	*VPU_DBG_REG3*	Debug register #3
0x000001B0	RW	0x0	*RET_VCORE0_CFG*	Configuration Information of V-CORE0
0x000001B4	RW	0x0	*RET_VCORE1_CFG*	Configuration Information of VCORE1
0x000001B8	RW	0x0	*RET_VCORE2_CFG*	Configuration Information of VCORE2
0x000001BC	RW	0x0	*RET_VCORE3_CFG*	Configuration Information of VCORE3
0x000001D0	RW	0x0	*VPU_RET_VCORE_PRESET*	Number of VCOREs present

1.2.4.2. 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.

1.2.4.2.1. INIT_VPU Command Parameter Registers

Table 1.5. INIT_VPU Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_INIT_OPTION*	INIT_VPU command option
0x00000110	RW	0x0	*CMD_ADDR_CODE_BASE*	Code Buffer Base Address
0x00000114	RW	0x0	*CMD_INIT_CODE_SIZE*	Code Buffer Size
0x00000118	RW	0x0	*CMD_INIT_CODE_PARAM*	Code Buffer Param
0x0000011C	RW	0x0	*CMD_INIT_ADDR_TEMP_BASE*	Temporal Buffer Base Address
0x00000120	RW	0x0	*CMD_INIT_TEMP_SIZE*	Temporal Buffer Size
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 Hw 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

Offset	Type	Reset Value	Name	Description
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
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason

1.2.4.2.2. WAKEUP_VPU Command Parameter Registers

Table 1.6. WAKEUP_VPU Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_WAKEUP_OPTION*	WAKEUP_VPU command option
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 Param
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 Hw 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
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
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason

1.2.4.2.3. SLEEP_VPU Command Parameter Registers

Table 1.7. SLEEP_VPU Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_SLEEP_OPTION*	SLEEP_VPU command option
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason

1.2.4.2.4. CREATE_INST Command Parameter Registers

Table 1.8. CREATE_INST Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_CREATE_INST_OPTION*	CREATE_INST command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
0x00000114	RW	0x0	*CMD_CREATE_INST_ADDR_WORK_BAS*	Work Buffer Base Address
0x00000118	RW	0x0	*CMD_CREATE_INST_WORK_SIZE*	Work Buffer Size
0x0000011C	RW	0x0	*CMD_CREATE_INST_BS_START_ADDR*	Bitstream Buffer Start Address
0x00000120	RW	0x0	*CMD_CREATE_INST_BS_SIZE*	Bitstream Buffer Size
0x00000124	RW	0x0	*CMD_CREATE_INST_BS_PARAM*	Bitstream Buffer Parameter
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason
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*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.5. FLUSH_INST Command Parameter Registers

Table 1.9. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_FLUSH_INST_OPTION*	FLUSH_INST command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason

Offset	Type	Reset Value	Name	Description
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*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.6. DESTROY_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_DESTROY_INST_OPTION*	DESTROY_INST command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason
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*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.7. INIT_SEQ 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_INIT_SEQ_OPTION*	INIT_SEQ command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
0x00000118	RW	0x0	*CMD_BS_RD_PTR*	Bistream Buffer Read Pointer
0x0000011C	RW	0x0	*CMD_BS_WR_PTR*	Bistream Buffer Write Pointer
0x00000120	RW	0x0	*CMD_BS_OPTIONS*	Bistream buffer option
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason
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_QUE_STATUS*	Queued command information
0x000001EC	RW	0x0	*RET_SEEK_INSTANCE_INFO*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.8. SET_FB Command Parameter Registers

Table 1.12. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_SET_FB_OPTION*	SET_FB command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
0x00000118	RW	0x0	*CMD_SET_FB_COMMON_PIC_INFO*	Dpb Information
0x00000118	RW	0x0	*CMD_SET_FB_STRIDE*	frame buffer stride
0x0000011C	RW	0x0	*CMD_SET_FB_PIC_SIZE*	Decoded Picture Size
0x00000120	RW	0x0	*CMD_SET_FB_SET_FB_NUM*	Set frame Number
0x00000120	RW	0x0	*SET_FB_INDICE*	SET_FB_INDICE
0x00000124	RW	0x0	*CMD_SET_FB_SCL_OUT_SIZE*	Scaler output size
0x00000134	RW	0x0	*CMD_SET_FB_ADDR_LUMA_BASE0*	Luma base of index0
0x00000134	RW	0x0	*CMD_SET_FB_ADDR_LUMA_BASE*	New luma base address to be replaced
0x00000138	RW	0x0	*CMD_SET_FB_ADDR_CB_BASE0*	Cb base of index0
0x00000138	RW	0x0	*CMD_SET_FB_ADDR_CB_BASE*	New Cb base address to be replaced
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
0x0000013C	RW	0x0	*CMD_SET_FB_ADDR_CR_BASE*	New Cr base address to be replaced
0x00000140	RW	0x0	*CMD_SET_FB_ADDR_FBC_C_OFFSET0*	FBC chroma offset base of index0
0x00000140	RW	0x0	*CMD_SET_FB_ADDR_MV_COL*	New colocated mv buffer base address to be replaced
0x00000144	RW	0x0	*CMD_SET_FB_ADDR_LUMA_BASE1*	Luma base of index1
0x00000144	RW	0x0	*CMD_SET_FB_ADDR_FBC_Y_BASE*	New FBC luma base address to be replaced
0x00000148	RW	0x0	*CMD_SET_FB_ADDR_CB_BASE1*	Cb base of index1
0x00000148	RW	0x0	*CMD_SET_FB_ADDR_FBC_C_BASE*	New FBC chroma base address to be replaced
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
0x0000014C	RW	0x0	*CMD_SET_FB_ADDR_FBC_Y_OFFSET*	New FBC luma offset base address to be replaced
0x00000150	RW	0x0	*CMD_SET_FB_ADDR_FBC_C_OFFSET1*	FBC chroma offset base of index1
0x00000150	RW	0x0	*CMD_SET_FB_ADDR_FBC_C_OFFSET*	New FBC chroma offset base address to be replaced
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

Offset	Type	Reset Value	Name	Description
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
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
0x000001C0	RW	0x0	CMD_SET_FB_ADDR_MV_COL3	Colocated mv buffer base of index 3
0x000001C4	RW	0x0	CMD_SET_FB_ADDR_MV_COL4	Colocated mv buffer base of index 4
0x000001C8	RW	0x0	CMD_SET_FB_ADDR_MV_COL5	Colocated mv buffer base of index 5
0x000001CC	RW	0x0	CMD_SET_FB_ADDR_MV_COL6	Colocated mv buffer base of index 6
0x000001D0	RW	0x0	CMD_SET_FB_ADDR_MV_COL7	Colocated mv buffer base of index 7
OUTPUT RETURN				
0x00000108	RW	0x0	RET_SUCCESS	Run Command Status
0x0000010C	RW	0x0	RET_FAIL_REASON	Fail Reason
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	Instance information in seeking phase
0x000001F0	RW	0x0	RET_PARSING_INSTANCE_INFO	Instance information in parsing phase
0x000001F4	RW	0x0	RET_DECODING_INSTANCE_INFO	Instance information in decoding phase
0x000001FC	RW	0x0	RET_DONE_INSTANCE_INFO	Instance information that has been done decoding

1.2.4.2.9. DEC_PIC Command Parameter Registers

Table 1.13. DEC_PIC Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	COMMAND	Run Command
0x00000104	RW	0x0	CMD_DEC_PIC_OPTION	DEC_PIC command option
0x00000110	RW	0x0	CMD_INSTANCE_INFO	Instance information
0x00000118	RW	0x0	CMD_BS_RD_PTR	Bistream Buffer Read Pointer
0x0000011C	RW	0x0	CMD_BS_WR_PTR	Bistream Buffer Write Pointer

Offset	Type	Reset Value	Name	Description
0x00000120	RW	0x0	*CMD_BS_OPTIONS*	Bistream buffer option
0x00000124	RW	0x0	*CMD_DEC_VCORE_LIMIT*	V-Core Limit
0x00000128	RW	0x0	*CMD_SEQ_CHANGE_ENABLE_FLAG*	Sequence change flag
0x0000012C	RW	0x0	*CMD_DEC_SEI_MASK*	User Data Mask
0x00000130	RW	0x0	*CMD_DEC_TEMPORAL_ID_PLUS1*	Max Decode Temporal ID
0x00000134	RW	0x0	*CMD_DEC_FORCE_FB_LATENCY_PLUS1*	User define display latency
0x00000150	RW	0x0	*CMD_DEC_USE_SEC_AXI*	Secondary AXI Usage
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason
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*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.10. QUERY Command Parameter Registers

1.2.4.2.10.1. GET_VPU_INFO

Table 1.14. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_QUERY_OPTION*	QUERY command option
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason
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 Date
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
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*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase

Offset	Type	Reset Value	Name	Description
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.10.2. GET_RESULT

Table 1.15. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_QUERY_OPTION*	QUERY command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
0x00000114	RW	0x0	*CMD_DEC_ADDR_USER_BASE*	User Data Buffer Base Address
0x00000118	RW	0x0	*CMD_DEC_USER_SIZE*	User Data Buffer Size
0x0000011C	RW	0x0	*CMD_DEC_USER_PARAM*	User Data Buffer Parameter
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason
0x0000011C	RW	0x0	*RET_QUERY_DEC_BS_RD_PTR*	Bitstream buffer read pointer
0x00000120	RW	0x0	*RET_QUERY_DEC_SEQ_PARAM*	Profile/Level/Tier/Max Sub layer
0x00000124	RW	0x0	*RET_DEC_COLOR_SAMPLE_INFO*	Color Sample Information
0x00000128	RW	0x0	*RET_DEC_ASPECT_RATIO*	Sample Aspect Ratio
0x0000012C	RW	0x0	*RET_DEC_BIT_RATE*	Maximum Bit Rate
0x00000130	RW	0x0	*RET_DEC_FRAME_RATE_NR*	Frame Rate Numerator
0x00000134	RW	0x0	*RET_DEC_FRAME_RATE_DR*	Frame Rate Denominator
0x00000138	RW	0x0	*RET_QUERY_DEC_NUM_REQURED_FB*	Required Number of Minimum DPB
0x0000013C	RW	0x0	*RET_QUERY_DEC_NUM_REORDER_DELAY*	Required Number of Minimum DPB
0x00000140	RW	0x0	*RET_QUERY_DEC_SUB_LAYER_INFO*	Number of Used V-CORE
0x00000144	RW	0x0	*RET_QUERY_DEC_NOTIFICATION*	Sequence change flag
0x00000148	RW	0x0	*RET_QUERY_DEC_USERDATA_IDC*	User data flag
0x0000014C	RW	0x0	*RET_QUERY_DEC_PIC_SIZE*	Decoded Picture Size
0x00000150	RW	0x0	*RET_QUERY_DEC_CROP_TOP_BOTTOM*	Display Crop Offset Top/Bottom
0x00000154	RW	0x0	*RET_QUERY_DEC_CROP_LEFT_RIGHT*	Display Crop Offset Left/Right
0x00000158	RW	0x0	*RET_QUERY_DEC_AU_START_POS*	AU Bitstream Start Position
0x0000015C	RW	0x0	*RET_QUERY_DEC_AU_END_POS*	AU Bitstream End Position
0x00000160	RW	0x0	*RET_QUERY_DEC_PIC_TYPE*	Decoded picture type
0x00000164	RW	0x0	*RET_QUERY_DEC_PIC_POC*	Picture Order Count
0x00000168	RW	0x0	*RET_QUERY_DEC_RECOVERY_POINT*	Recovery point
0x0000016C	RW	0x0	*RET_QUERY_DEC_DEBUG_INDEX*	Decoded picture index of DPB
0x00000170	RW	0x0	*RET_QUERY_DEC_DECODED_INDEX*	Decoded picture index of DPB
0x00000174	RW	0x0	*RET_QUERY_DEC_DISPLAY_INDEX*	Display picture index of DPB
0x00000178	RW	0x0	*RET_QUERY_DEC_REALLOC_INDEX*	Display picture index of DPB
0x0000017C	RW	0x0	*RET_QUERY_DEC_DISP_IDC*	Display flag
0x00000180	RW	0x0	*RET_QUERY_DEC_NUM_ERR_CTB*	Number of error CTU
0x000001D4	RW	0x0	*RET_QUERY_DEC_SEEK_CYCLE*	Seek cycle
0x000001D8	RW	0x0	*RET_QUERY_DEC_PARSING_CYCLE*	Parsing cycle
0x000001DC	RW	0x0	*RET_QUERY_DEC_DECODING_CYCLE*	Decoding cycle
0x000001E0	RW	0x0	*RET_QUERY_DEC_FRAME_CYCLE*	Frame cycle

Offset	Type	Reset Value	Name	Description
0x000001E4	RW	0x0	*RET_DEC_WARN_INFO*	Warning information
0x000001E4	RW	0x0	*RET_BS_EMPTY*	Bitstream buffer empty flag
0x000001E8	RW	0x0	*RET_DEC_ERR_INFO*	Error information
0x000001E8	RW	0x0	*RET_QUEUED_CMD_DONE*	Queued command done flag
0x000001EC	RW	0x0	*RET_QUERY_DEC_SUCCESS*	Query result
0x000001EC	RW	0x0	*RET_SEEK_INSTANCE_INFO*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.10.3. UPDATE_DISP_IDC

Table 1.16. Register summary

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_QUERY_OPTION*	QUERY command option
0x00000110	RW	0x0	*CMD_INSTANCE_INFO*	Instance information
0x00000118	RW	0x0	*CMD_QUERY_DEC_SET_DISP_IDC*	CMD_QUERY_DEC_SET_DISP_IDC
0x0000011C	RW	0x0	*CMD_QUERY_DEC_CLR_DISP_IDC*	CMD_QUERY_DEC_CLR_DISP_IDC
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run Command Status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail Reason
0x0000017C	RW	0x0	*RET_QUERY_DEC_DISP_IDC*	RET_QUERY_DEC_DISP_IDC
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*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.4.2.11. UPDATE_BS Parameter Registers

Table 1.17. UPDATE_BS Parameter Registers

Offset	Type	Reset Value	Name	Description
INPUT ARGUMENT				
0x00000100	RW	0x0	*COMMAND*	Run Command
0x00000104	RW	0x0	*CMD_UPDATE_BS_OPTION*	UPDATE_BS command option
0x0000011C	RW	0x0	*CMD_BS_WR_PTR*	Bistream buffer write pointer
0x00000120	RW	0x0	*CMD_BS_OPTIONS*	Bistream buffer option
OUTPUT RETURN				
0x00000108	RW	0x0	*RET_SUCCESS*	Run command status
0x0000010C	RW	0x0	*RET_FAIL_REASON*	Fail reason
0x000001E4	RW	0x0	*RET_BS_EMPTY*	Bitstream buffer empty flag
0x000001E8	RW	0x0	*RET_QUEUED_CMD_DONE*	Queued command done flag

Offset	Type	Reset Value	Name	Description
0x000001EC	RW	0x0	*RET_SEEK_INSTANCE_INFO*	Instance information in seeking phase
0x000001F0	RW	0x0	*RET_PARSING_INSTANCE_INFO*	Instance information in parsing phase
0x000001F4	RW	0x0	*RET_DECODING_INSTANCE_INFO*	Instance information in decoding phase
0x000001FC	RW	0x0	*RET_DONE_INSTANCE_INFO*	Instance information that has been done decoding

1.2.5. 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.

1.2.5.1. Control Register Descriptions

1.2.5.1.1. VPU_PO_CONF (0x00000000)

Power On Configurations

Table 1.18. 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 1.19. 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

1.2.5.1.2. VCPU_CUR_PC (0x00000004)

Current program counter value of V-CPU

Table 1.20. 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 1.21. 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

1.2.5.1.3. VCPU_CUR_LR (0x00000008)

Current LR (for debugger only)

Table 1.22. 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 1.23. 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

1.2.5.1.4. VPU_PDBG_STEP_MASK (0x0000000C)

Debugger control
(for debugger only)

Table 1.24. 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 1.25. 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

1.2.5.1.5. VPU_PDBG_CTRL (0x00000010)

Debugger control
(for debugger only)

Table 1.26. 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 1.27. 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

1.2.5.1.6. VPU_PDBG_IDX_REG (0x00000014)

V-CPU debugger index register
(For debugger only)

Table 1.28. 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 1.29. 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

1.2.5.1.7. VPU_PDBG_WDATA_REG (0x00000018)

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

Table 1.30. 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 1.31. 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

1.2.5.1.8. VPU_PDBG_RDATA_REG (0x0000001C)

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

Table 1.32. 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 1.33. 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

1.2.5.1.9. 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 1.34. 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 1.35. 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

1.2.5.1.10. VPU_FIO_DATA (0x00000024)

FIO data

Table 1.36. 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 1.37. 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

1.2.5.1.11. 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 1.38. 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	RSVD	RSVD	RSVD	RSVD	RSVD	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 1.39. 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]	RSVD	RW	Reserved	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	RSVD	RW	Reserved	0x0
[8]	CMD8_INTR_USER	RW	DEC_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

1.2.5.1.12. 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 1.40. 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	RSVD	RSVD	RSVD	RSVD	RSVD	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 1.41. 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]	RSVD	RW	Reserved	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	RSVD	RW	Reserved	0x0
[8]	CMD8_CLR	RW	DEC_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

1.2.5.1.13. VPU_HOST_INT_REQ (0x00000038)

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

Table 1.42. 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 1.43. 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

1.2.5.1.14. VPU_VINT_CLEAR (0x0000003C)

VPU interrupt clear

Table 1.44. 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 1.45. 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

1.2.5.1.15. VPU_HINT_CLEAR (0x00000040)

Host interrupt clear

Table 1.46. 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 1.47. 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

1.2.5.1.16. VPU_VPU_INT_STS (0x00000044)

Interrupt Status

Table 1.48. 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 1.49. 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

1.2.5.1.17. VPU_VINT_ENABLE (0x00000048)

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

Table 1.50. 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	RSVD	RSVD	RSVD	RSVD	RSVD	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 1.51. 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]	RSVD	RW	Reserved	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	RSVD	RW	Reserved	0x0
[8]	CMD8_EN	RW	DEC_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

1.2.5.1.18. 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 1.52. 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	RSVD	RSVD	RSVD	RSVD	RSVD	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 1.53. 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]	RSVD	RW	Reserved	0x0
[12]	RSVD	RW	Reserved	0x0
[11]	RSVD	RW	Reserved	0x0
[10]	RSVD	RW	Reserved	0x0
[9]	RSVD	RW	Reserved	0x0
[8]	CMD8_INTR	RW	DEC_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

1.2.5.1.19. VPU_RESET_REQ (0x00000050)

VPU reset request for each block of clock domain
0: reset request clear or do nothing
1: reset request For more details, please refer to *clock and reset signals* section in datasheet.

Table 1.54. 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						VRST_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	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

Table 1.55. VPU_RESET_REQ Field Description

Bit	Name	Type	Function	Reset Value
[31:26]	RSVD	R	Reserved	0x0
[25]	VRST_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 V-CORE) reset request [23-20] Reserved [19] : V-CORE3 [18] : V-CORE2 [17] : V-CORE1	0x0

Bit	Name	Type	Function	Reset Value
			[16] : V-CORE0	
[15:8]	BRST_REQ	RW	BCLK domain (for each V-CORE) 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 V-CORE) reset request [7:4]: Reserved [3] : V-CORE3 [2] : V-CORE2 [1] : V-CORE1 [0] : V-CORE0	0x0

1.2.5.1.20. VPU_RESET_STATUS (0x00000054)

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

Table 1.56. 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						VRST_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	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

Table 1.57. VPU_RESET_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:26]	RSVD	R	Reserved	0x0
[25]	VRST_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 V-CORE) reset status	0x0
[15:8]	BRST_STS	RO	BCLK domain (for each V-CORE) reset status	0x0
[7:0]	CRST_STS	RO	CCLK domain (for each V-CORE) reset status	0x0

1.2.5.1.21. VCPU_RESTART (0x00000058)

V-CPU restart request

Table 1.58. 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 1.59. 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

1.2.5.1.22. VPU_CLK_MASK (0x0000005C)

Clock gating control register for each clock domain

Table 1.60. 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						VCLK_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	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

Table 1.61. VPU_CLK_MASK Field Description

Bit	Name	Type	Function	Reset Value
[31:26]	RSVD	R	Reserved	0x0
[25]	VCLK_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

1.2.5.1.23. 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 1.62. 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 1.63. 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

1.2.5.1.24. VPU_REMAP_VADDR (0x00000064)

Remap region base address in virtual address space

Table 1.64. 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 1.65. 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. 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	0x0

Bit	Name	Type	Function	Reset Value
			V-CPU always start booting up from virtual address 0. If not, VPU can not boot-up.	
[11:0]	RSVD	R	Reserved	0x0

1.2.5.1.25. VPU_REMAP_PADDR (0x00000068)

Remap region base address in physical address space

Table 1.66. VPU_REMAP_PADDR Bit Assignment

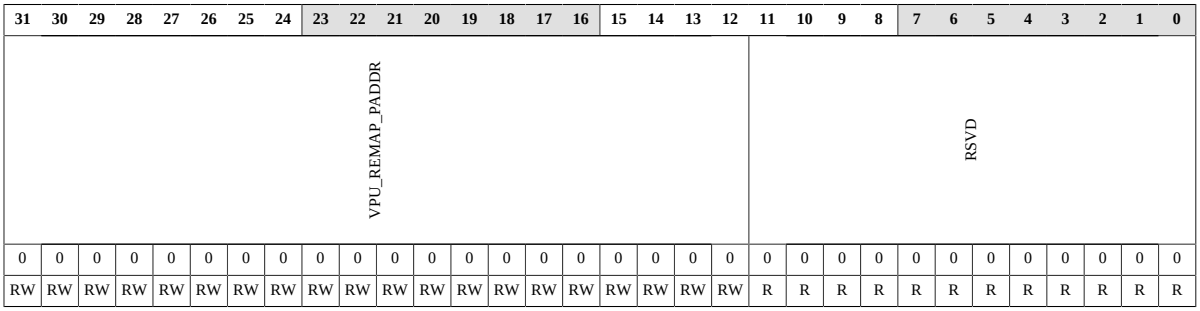


Table 1.67. VPU_REMAP_PADDR Field Description

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

1.2.5.1.26. VPU_REMAP_CORE_START (0x0000006C)

VPU Start Request

Table 1.68. VPU_REMAP_CORE_START Bit Assignment

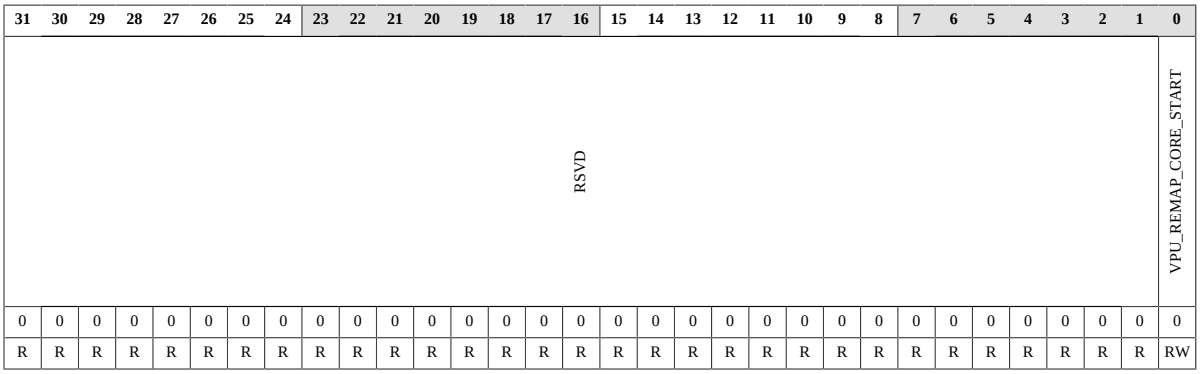


Table 1.69. VPU_REMAP_CORE_START Field Description

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. After setting up remap or initial configuartion, host should set this register to init VPU.	0x0

1.2.5.1.27. VPU_BUSY_STATUS (0x00000070)

VPU Busy Status

Table 1.70. VPU_BUSY_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_BUSY_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	RW

Table 1.71. 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

1.2.5.1.28. VPU_HALT_STATUS (0x00000074)

For power control, status checking of V-CPU (Not implemented yet)

Table 1.72. VPU_HALT_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_HALT_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	RO	RO	RO	RO	RO

Table 1.73. VPU_HALT_STATUS Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	VPU_HALT_STATUS	RO	0x1 : V-CPU core halt	0x0

1.2.5.1.29. VPU_VCPU_STATUS (0x00000078)

V-CPU bus busy and V-CPU status

Table 1.74. 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

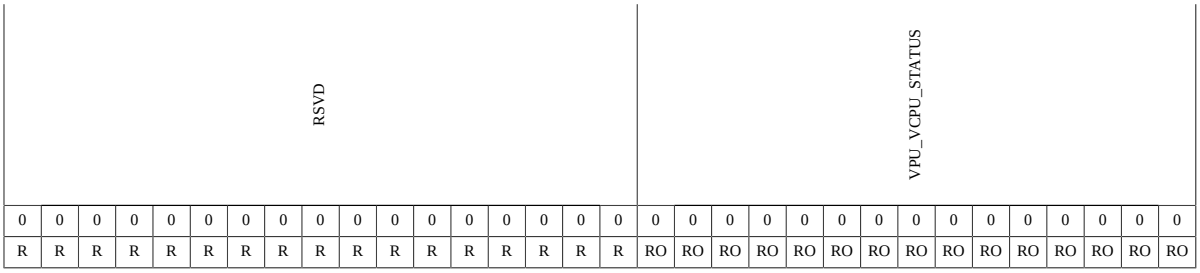


Table 1.75. VPU_VCPU_STATUS Field Description

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

1.2.5.1.30. VPU_PRESCAN_STATUS (0x0000007C)

Table 1.76. VPU_PRESCAN_STATUS Bit Assignment

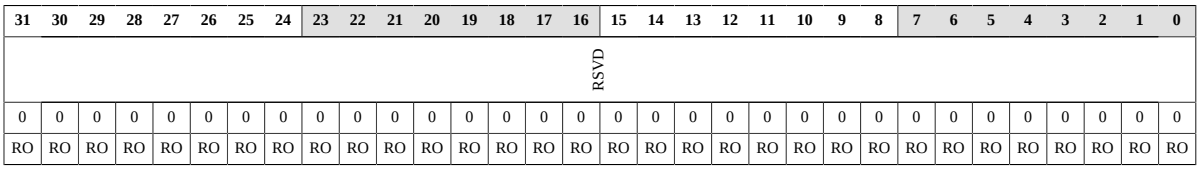


Table 1.77. VPU_PRESCAN_STATUS Field Description

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

1.2.5.1.31. RET_FIO_STATUS (0x00000080)

Table 1.78. RET_FIO_STATUS Bit Assignment

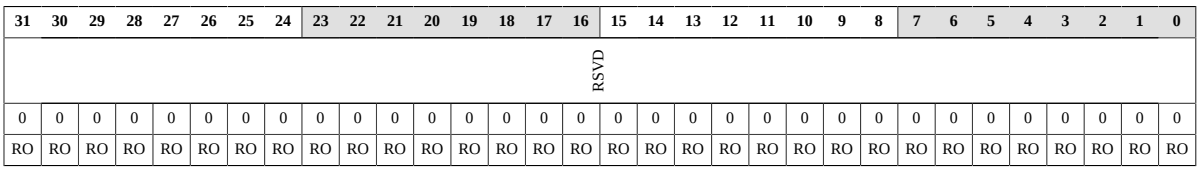
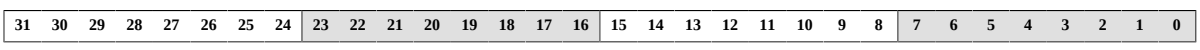


Table 1.79. RET_FIO_STATUS Field Description

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

1.2.5.1.32. RET_PRODUCT_NAME (0x00000090)

Table 1.80. RET_PRODUCT_NAME Bit Assignment



RSVD																												HW_NAME			
0	0	0	0	0	0	0	0	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 1.81. 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 always returns "WAVE" for WAVE4x, WAVE5x product series	0x0

1.2.5.1.33. RET_PRODUCT_VERSION (0x00000094)

Table 1.82. RET_PRODUCT_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
RSVD																												HW_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 1.83. 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 always returns "5120" for WAVE512 product.	0x0

1.2.5.1.34. RET_VCPU_CONFIG0 (0x00000098)

Table 1.84. RET_VCPU_CONFIG0 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 1.85. RET_VCPU_CONFIG0 Field Description

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

1.2.5.1.35. RET_VCPU_CONFIG1 (0x0000009C)

Table 1.86. 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 1.87. RET_VCPU_CONFIG1 Field Description

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

1.2.5.1.36. RET_CODEC_STD (0x000000A0)

Table 1.88. 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 1.89. RET_CODEC_STD Field Description

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

1.2.5.1.37. RET_CONF_DATE (0x000000A4)

Table 1.90. 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 1.91. 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

1.2.5.1.38. RET_CONF_REVISION (0x000000A8)

Table 1.92. 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 1.93. 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

1.2.5.1.39. RET_CONF_TYPE (0x000000AC)

Table 1.94. 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 1.95. 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

1.2.5.1.40. VPU_DBG_REG0 (0x000000F0)

Table 1.96. VPU_DBG_REG0 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
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 1.97. VPU_DBG_REG0 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	RW	Debug register #0	0x0

1.2.5.1.41. VPU_DBG_REG1 (0x000000F4)

Table 1.98. VPU_DBG_REG1 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
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 1.99. VPU_DBG_REG1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	RW	Debug register #1	0x0

1.2.5.1.42. VPU_DBG_REG2 (0x000000F8)

Table 1.100. VPU_DBG_REG2 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

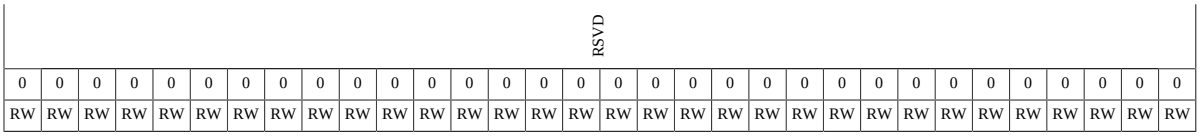


Table 1.101. VPU_DBG_REG2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	RW	Debug register #2	0x0

1.2.5.1.43. VPU_DBG_REG3 (0x000000FC)

Table 1.102. VPU_DBG_REG3 Bit Assignment

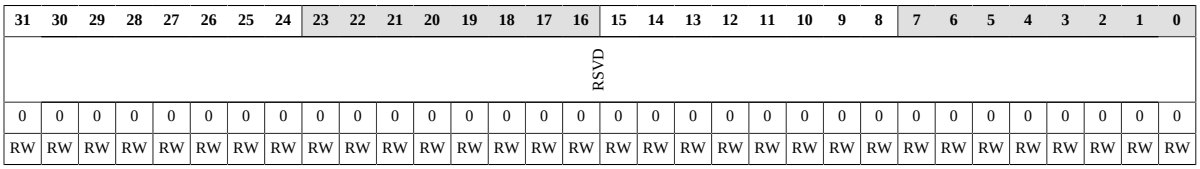


Table 1.103. VPU_DBG_REG3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RSVD	RW	Debug register #3	0x0

1.2.5.1.44. RET_VCORE0_CFG (0x000001B0)

Table 1.104. RET_VCORE0_CFG Bit Assignment

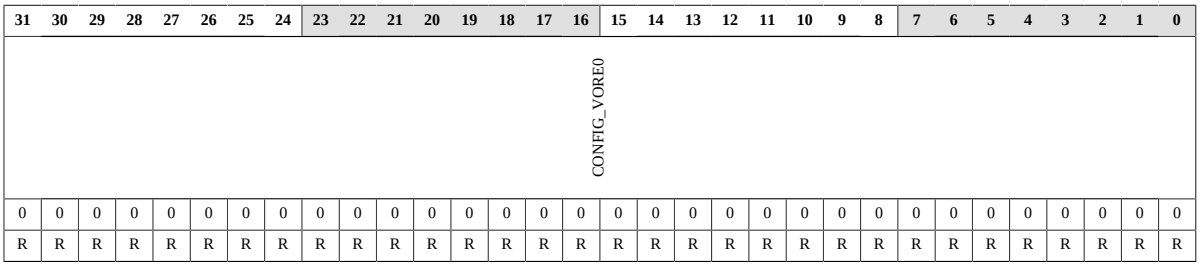
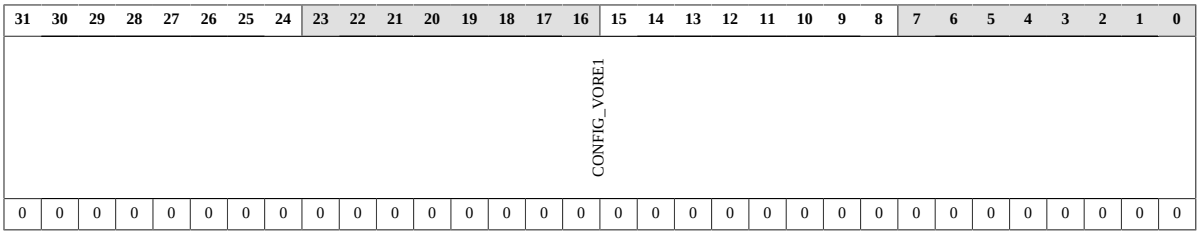


Table 1.105. RET_VCORE0_CFG Field Description

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

1.2.5.1.45. RET_VCORE1_CFG (0x000001B4)

Table 1.106. RET_VCORE1_CFG 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 1.107. RET_VCORE1_CFG Field Description

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

1.2.5.1.46. RET_VCORE2_CFG (0x000001B8)

Table 1.108. RET_VCORE2_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_VORE2																															
0	0	0	0	0	0	0	0	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 1.109. RET_VCORE2_CFG Field Description

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

1.2.5.1.47. RET_VCORE3_CFG (0x000001BC)

Table 1.110. RET_VCORE3_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_VORE3																															
0	0	0	0	0	0	0	0	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 1.111. RET_VCORE3_CFG Field Description

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

1.2.5.1.48. VPU_RET_VCORE_PRESET (0x000001D0)

Table 1.112. VPU_RET_VCORE_PRESET 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
VCORE_PRESENT																															
0	0	0	0	0	0	0	0	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 1.113. VPU_RET_VCORE_PRESET Field Description

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

1.2.5.2. Command I/O Register Descriptions

1.2.5.2.1. 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.

1.2.5.2.1.1. 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 **INIT_VPU** command (**0x0001** is given to this register).

Table 1.114. 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 1.115. 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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.1.2. CMD_INIT_OPTION (0x00000104)

Reserved

Table 1.116. CMD_INIT_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
INIT_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
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 1.117. CMD_INIT_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	INIT_OPTION	RW	Reserved	0x0

1.2.5.2.1.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.118. 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 1.119. 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_WARNIG	0x0

1.2.5.2.1.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.120. 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 1.121. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.1.5. CMD_ADDR_CODE_BASE (0x00000110)

Code Buffer Base Address

Table 1.122. CMD_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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/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

Table 1.123. CMD_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

1.2.5.2.1.6. CMD_INIT_CODE_SIZE (0x00000114)

Code Buffer Size

Table 1.124. CMD_INIT_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 1.125. CMD_INIT_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

1.2.5.2.1.7. CMD_INIT_CODE_PARAM (0x00000118)

Parameter for code buffer

Table 1.126. CMD_INIT_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 1.127. CMD_INIT_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

Bit	Name	Type	Function	Reset Value
[3:0]	CODE_ENDIAN	R/W	Endianness	0x0

1.2.5.2.1.8. CMD_INIT_ADDR_TEMP_BASE (0x0000011C)

Temporal Buffer Base Address

Table 1.128. CMD_INIT_ADDR_TEMP_BASE Bit Assignment

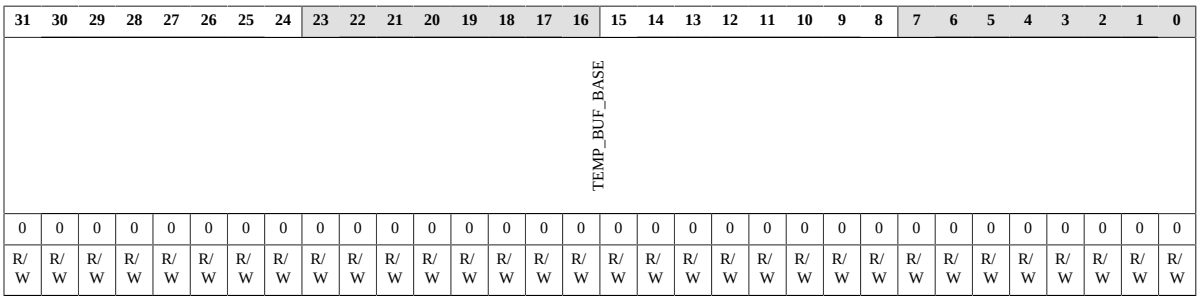


Table 1.129. CMD_INIT_ADDR_TEMP_BASE Field Description

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

1.2.5.2.1.9. CMD_INIT_TEMP_SIZE (0x00000120)

Temporal Buffer Size

Table 1.130. CMD_INIT_TEMP_SIZE Bit Assignment

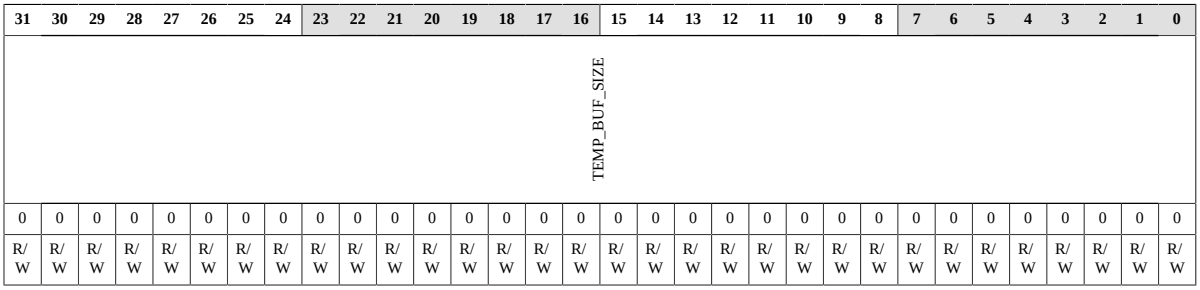


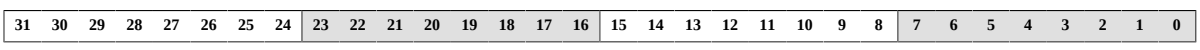
Table 1.131. CMD_INIT_TEMP_SIZE Field Description

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

1.2.5.2.1.10. CMD_INIT_ADDR_SEC_AXI (0x00000124)

Secondary AXI Base Address
It is valid only when USE_SEC_AXI is not 0.

Table 1.132. CMD_INIT_ADDR_SEC_AXI Bit Assignment



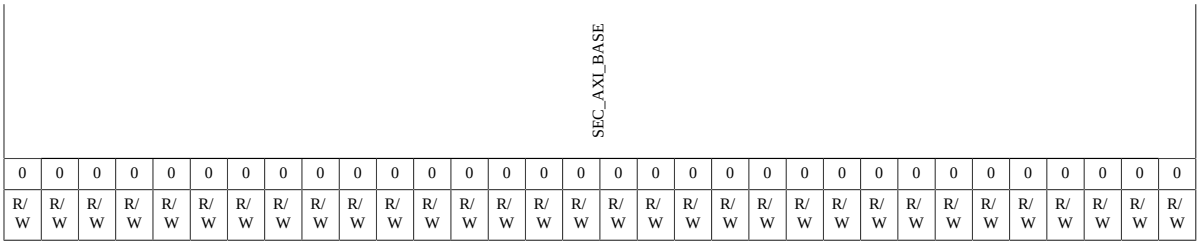


Table 1.133. CMD_INIT_ADDR_SEC_AXI Field Description

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

1.2.5.2.1.11. CMD_INIT_SEC_AXI_SIZE (0x00000128)

Seconary AXI Memory Size
It is valid only when USE_SEC_AXI is not 0.

Table 1.134. CMD_INIT_SEC_AXI_SIZE Bit Assignment

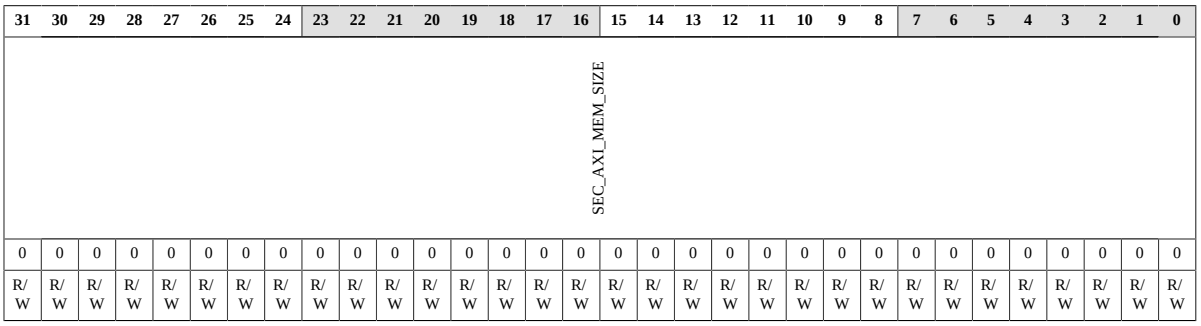


Table 1.135. CMD_INIT_SEC_AXI_SIZE Field Description

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

1.2.5.2.1.12. CMD_INIT_HW_OPTION (0x0000012C)

VPU hardware options

Table 1.136. CMD_INIT_HW_OPTION Bit Assignment

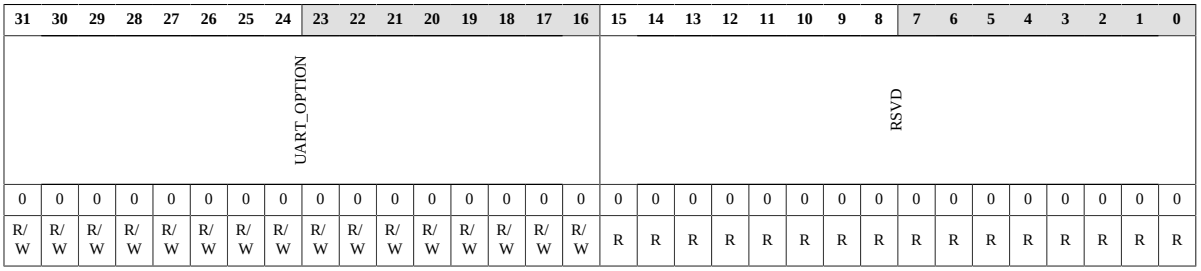


Table 1.137. CMD_INIT_HW_OPTION Field Description

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

1.2.5.2.1.13. CMD_WAKEUP_SYSTEM_CLOCK (0x00000130)

Reserved

Table 1.138. CMD_WAKEUP_SYSTEM_CLOCK 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
SYSTEM_CLOCK																															
0	0	0	0	0	0	0	0	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. CMD_WAKEUP_SYSTEM_CLOCK Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SYSTEM_CLOCK	R/W	The maximum number of cycle information that is allowed for watchdog timer	0x0

1.2.5.2.1.14. CMD_INIT_NUM_TASK_BUF (0x00000134)

Table 1.140. CMD_INIT_NUM_TASK_BUF 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																											NUM_TASK_BUF				
0	0	0	0	0	0	0	0	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 1.141. CMD_INIT_NUM_TASK_BUF Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	NUM_TASK_BUF	R/W	The number of valid task buffer (for max queueing depth)	0x0

1.2.5.2.1.15. CMD_INIT_ADDR_TASK_BUF0 (0x00000138)

Table 1.142. 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 1.143. 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

1.2.5.2.1.16. CMD_INIT_ADDR_TASK_BUF1 (0x0000013C)

Table 1.144. 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 1.145. CMD_INIT_ADDR_TASK_BUF1 Field Description

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

1.2.5.2.1.17. CMD_INIT_ADDR_TASK_BUF2 (0x00000140)

Table 1.146. 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 1.147. 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

1.2.5.2.1.18. CMD_INIT_ADDR_TASK_BUF3 (0x00000144)

Table 1.148. 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 1.149. 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

1.2.5.2.1.19. CMD_INIT_ADDR_TASK_BUF4 (0x00000148)

Table 1.150. CMD_INIT_ADDR_TASK_BUF4 Bit Assignment

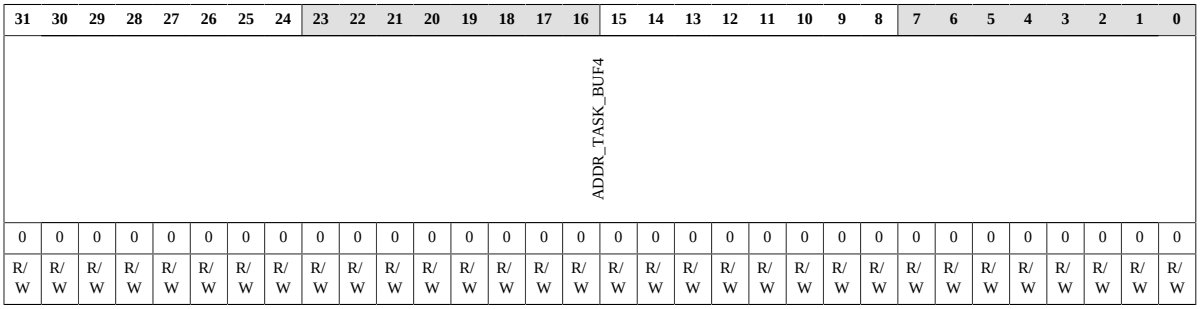


Table 1.151. CMD_INIT_ADDR_TASK_BUF4 Field Description

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

1.2.5.2.1.20. CMD_INIT_ADDR_TASK_BUF5 (0x0000014C)

Table 1.152. CMD_INIT_ADDR_TASK_BUF5 Bit Assignment

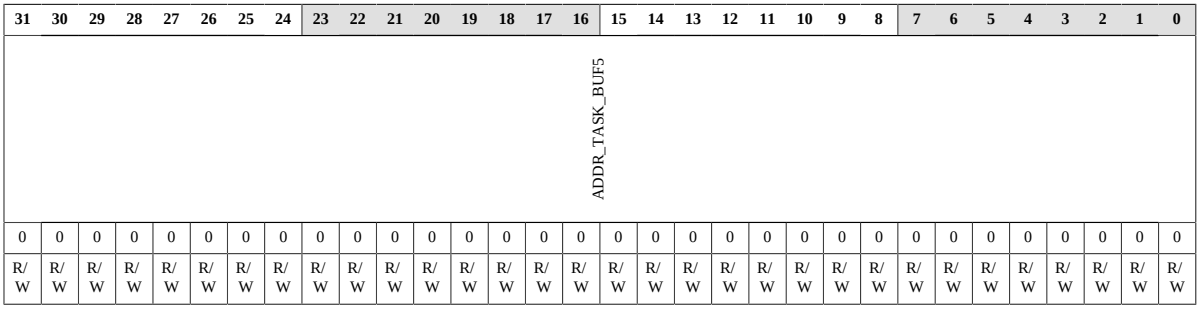


Table 1.153. CMD_INIT_ADDR_TASK_BUF5 Field Description

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

1.2.5.2.1.21. CMD_INIT_ADDR_TASK_BUF6 (0x00000150)

Table 1.154. CMD_INIT_ADDR_TASK_BUF6 Bit Assignment

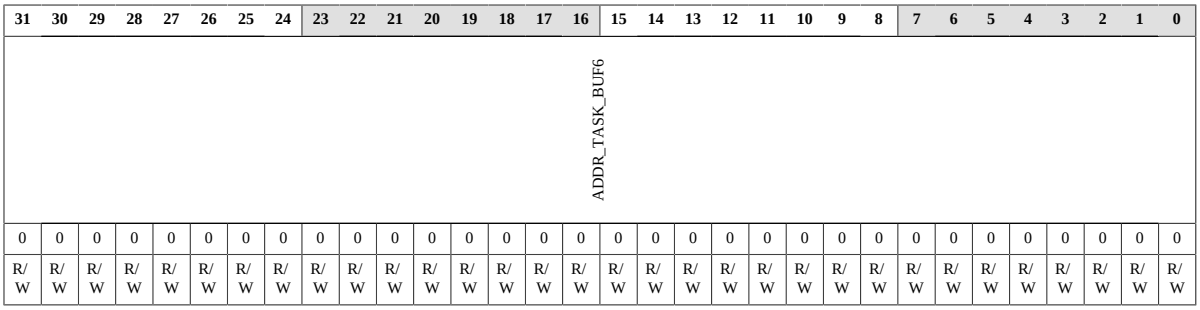


Table 1.155. CMD_INIT_ADDR_TASK_BUF6 Field Description

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

1.2.5.2.1.22. CMD_INIT_ADDR_TASK_BUF7 (0x00000154)

Table 1.156. 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 1.157. 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

1.2.5.2.1.23. CMD_INIT_ADDR_TASK_BUF8 (0x00000158)

Table 1.158. 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 1.159. 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

1.2.5.2.1.24. CMD_INIT_ADDR_TASK_BUF9 (0x0000015C)

Table 1.160. 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 1.161. 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

1.2.5.2.1.25. CMD_INIT_ADDR_TASK_BUFA (0x00000160)

Table 1.162. 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 1.163. 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

1.2.5.2.1.26. CMD_INIT_ADDR_TASK_BUFB (0x00000164)

Table 1.164. 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 1.165. 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

1.2.5.2.1.27. CMD_INIT_ADDR_TASK_BUFC (0x00000168)

Table 1.166. 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 1.167. 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

1.2.5.2.1.28. CMD_INIT_ADDR_TASK_BUFD (0x0000016C)

Table 1.168. 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 1.169. 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

1.2.5.2.1.29. CMD_INIT_ADDR_TASK_BUFE (0x00000170)

Table 1.170. 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 1.171. 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

1.2.5.2.1.30. CMD_INIT_ADDR_TASK_BUFF (0x00000174)

Table 1.172. 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 1.173. 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

1.2.5.2.2. 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.

1.2.5.2.2.1. 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 **WAKEUP_VPU** command (**0x0002** is given to this register).

Table 1.174. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.2.2. CMD_WAKEUP_OPTION (0x00000104)

Table 1.176. CMD_WAKEUP_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 1.177. CMD_WAKEUP_OPTION Field Description

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

1.2.5.2.2.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.178. 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 1.179. 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_WARNIG	0x0

1.2.5.2.2.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.180. 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 1.181. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.2.5. CMD_WAKEUP_ADDR_CODE_BASE (0x00000110)

Code Buffer Base Address

Table 1.182. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/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

Table 1.183. 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

1.2.5.2.2.6. CMD_WAKEUP_CODE_SIZE (0x00000114)

Code Buffer Size

Table 1.184. 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 1.185. 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

1.2.5.2.2.7. CMD_WAKEUP_CODE_PARAM (0x00000118)

Parameter for code buffer

Table 1.186. 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
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 1.187. 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

Bit	Name	Type	Function	Reset Value
[3:0]	CODE_ENDIAN	R/W	Endianness	0x0

1.2.5.2.2.8. CMD_WAKEUP_ADDR_TEMP_BASE (0x0000011C)

Temporal Buffer Base Address

Table 1.188. CMD_WAKEUP_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 1.189. 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

1.2.5.2.2.9. CMD_WAKEUP_TEMP_SIZE (0x00000120)

Temporal Buffer Size

Table 1.190. CMD_WAKEUP_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 1.191. 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

1.2.5.2.2.10. CMD_WAKEUP_ADDR_SEC_AXI (0x00000124)

Secondary AXI Base Address
It is valid only when USE_SEC_AXI is not 0.

Table 1.192. CMD_WAKEUP_ADDR_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

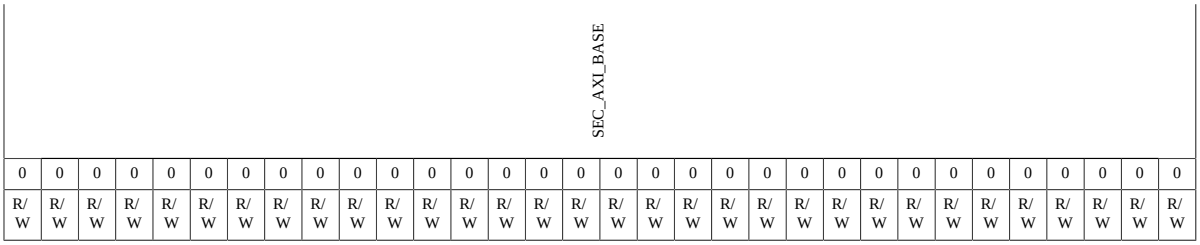


Table 1.193. CMD_WAKEUP_ADDR_SEC_AXI Field Description

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

1.2.5.2.2.11. CMD_WAKEUP_SEC_AXI_SIZE (0x00000128)

Seconary AXI Memory Size
It is valid only when USE_SEC_AXI is not 0.

Table 1.194. CMD_WAKEUP_SEC_AXI_SIZE Bit Assignment

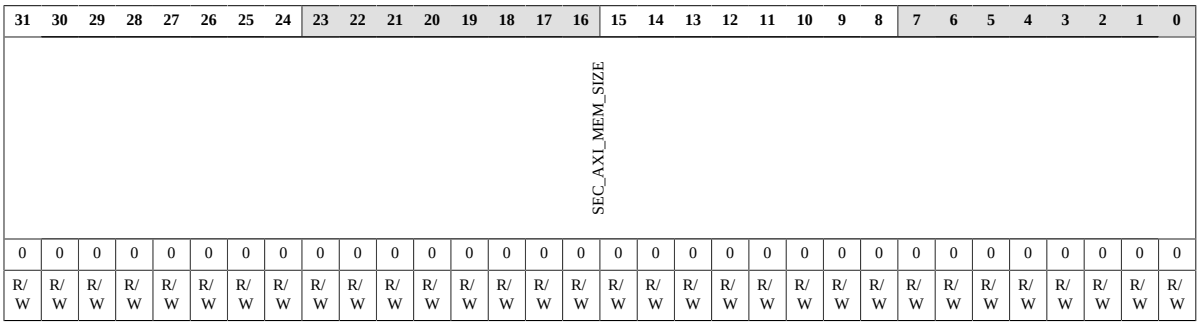


Table 1.195. CMD_WAKEUP_SEC_AXI_SIZE Field Description

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

1.2.5.2.2.12. CMD_WAKEUP_HW_OPTION (0x0000012C)

VPU hardware options

Table 1.196. CMD_WAKEUP_HW_OPTION Bit Assignment

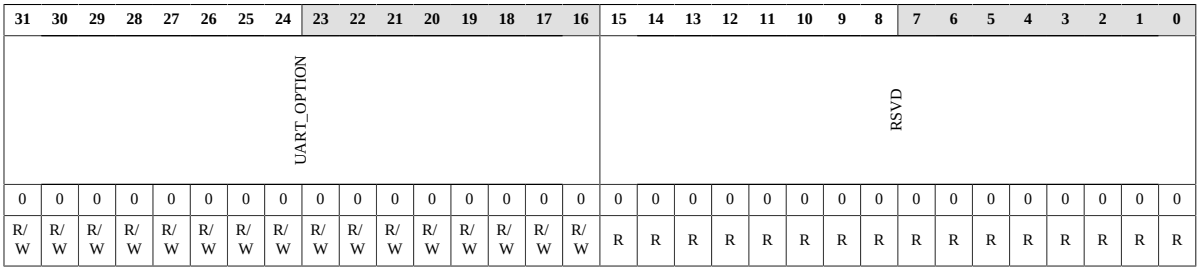


Table 1.197. CMD_WAKEUP_HW_OPTION Field Description

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

1.2.5.2.2.13. CMD_WAKEUP_SYSTEM_CLOCK (0x00000130)

Reserved

Table 1.198. CMD_WAKEUP_SYSTEM_CLOCK Bit Assignment

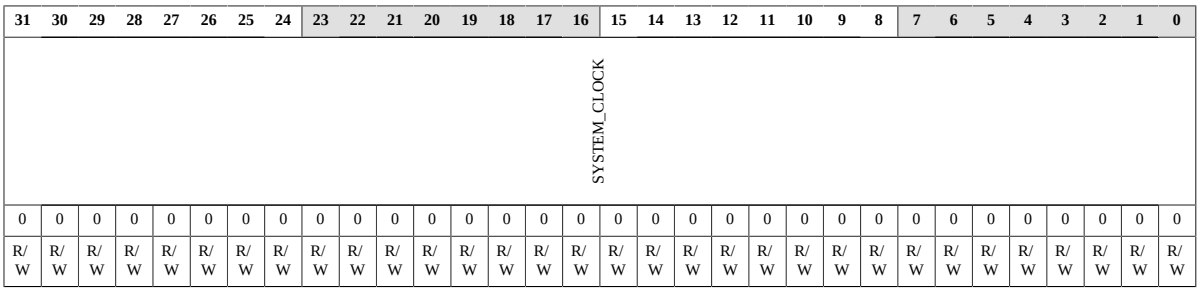


Table 1.199. CMD_WAKEUP_SYSTEM_CLOCK Field Description

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

1.2.5.2.2.14. CMD_WAKEUP_NUM_TASK_BUF (0x00000134)

Table 1.200. CMD_WAKEUP_NUM_TASK_BUF Bit Assignment

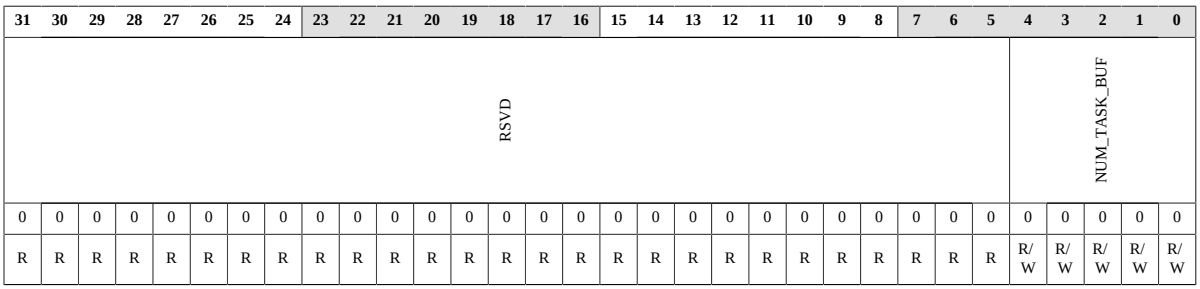


Table 1.201. CMD_WAKEUP_NUM_TASK_BUF Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	NUM_TASK_BUF	R/W	Number of valid task buffer (for max queueing depth)	0x0

1.2.5.2.2.15. CMD_WAKEUP_ADDR_TASK_BUF0 (0x00000138)

Table 1.202. CMD_WAKEUP_ADDR_TASK_BUF0 Bit Assignment

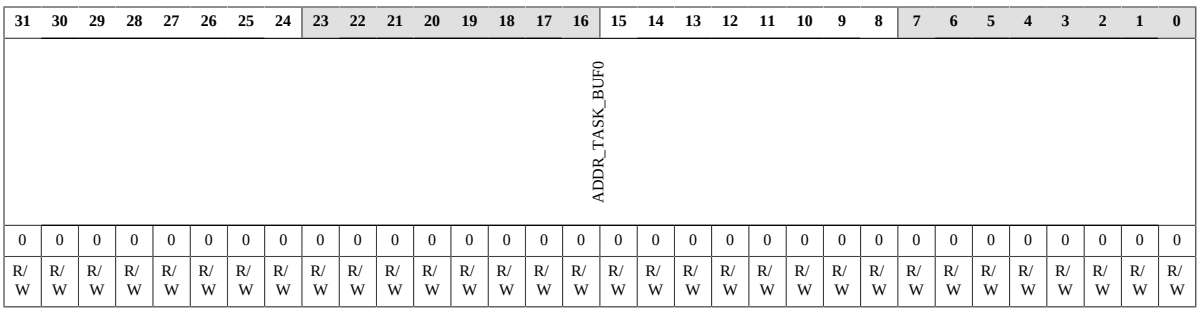


Table 1.203. CMD_WAKEUP_ADDR_TASK_BUF0 Field Description

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

1.2.5.2.2.16. CMD_WAKEUP_ADDR_TASK_BUF1 (0x0000013C)

Table 1.204. 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 1.205. 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

1.2.5.2.2.17. CMD_WAKEUP_ADDR_TASK_BUF2 (0x00000140)

Table 1.206. 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 1.207. 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

1.2.5.2.2.18. CMD_WAKEUP_ADDR_TASK_BUF3 (0x00000144)

Table 1.208. 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 1.209. 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

1.2.5.2.2.19. CMD_WAKEUP_ADDR_TASK_BUF4 (0x00000148)

Table 1.210. 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 1.211. 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

1.2.5.2.2.20. CMD_WAKEUP_ADDR_TASK_BUF5 (0x0000014C)

Table 1.212. 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 1.213. 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

1.2.5.2.2.21. CMD_WAKEUP_ADDR_TASK_BUF6 (0x00000150)

Table 1.214. 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 1.215. 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

1.2.5.2.2.22. CMD_WAKEUP_ADDR_TASK_BUF7 (0x00000154)

Table 1.216. 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 1.217. 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

1.2.5.2.2.23. CMD_WAKEUP_ADDR_TASK_BUF8 (0x00000158)

Table 1.218. 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 1.219. 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

1.2.5.2.2.24. CMD_WAKEUP_ADDR_TASK_BUF9 (0x0000015C)

Table 1.220. 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 1.221. 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

1.2.5.2.2.25. CMD_WAKEUP_ADDR_TASK_BUFA (0x00000160)

Table 1.222. 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 1.223. 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

1.2.5.2.2.26. CMD_WAKEUP_ADDR_TASK_BUFB (0x00000164)

Table 1.224. 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 1.225. 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

1.2.5.2.2.27. CMD_WAKEUP_ADDR_TASK_BUFC (0x00000168)

Table 1.226. CMD_WAKEUP_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 1.227. 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

1.2.5.2.2.28. CMD_WAKEUP_ADDR_TASK_BUFD (0x0000016C)

Table 1.228. CMD_WAKEUP_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 1.229. 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

1.2.5.2.2.29. CMD_WAKEUP_ADDR_TASK_BUFE (0x00000170)

Table 1.230. CMD_WAKEUP_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 1.231. 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

1.2.5.2.2.30. CMD_WAKEUP_ADDR_TASK_BUFF (0x00000174)

Table 1.232. 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 1.233. 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

1.2.5.2.3. SLEEP_VPU Command Parameter Registers

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

1.2.5.2.3.1. 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 **SLEEP_VPU** command (**0x0004** is given to this register).

Table 1.234. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.235. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.3.2. CMD_SLEEP_OPTION (0x00000104)

Table 1.236. CMD_SLEEP_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 1.237. CMD_SLEEP_OPTION Field Description

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

1.2.5.2.3.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.238. 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

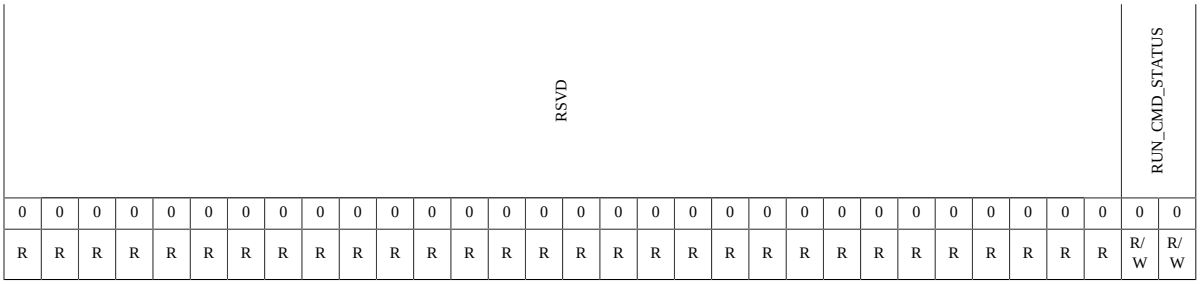


Table 1.239. RET_SUCCESS Field Description

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_WARNIG	0x0

1.2.5.2.3.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.240. RET_FAIL_REASON Bit Assignment

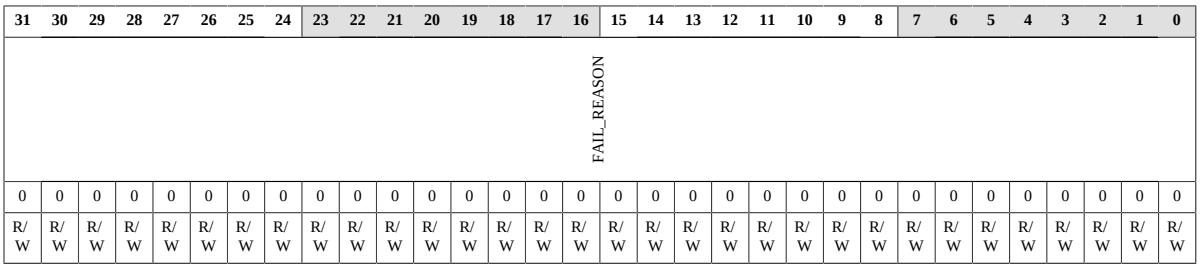


Table 1.241. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.4. CREATE_INST Command Parameter Registers

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

1.2.5.2.4.1. 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 **CREATE_INST** command (**0x0008** is given to this register).

Table 1.242. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.4.2. CMD_CREATE_INST_OPTION (0x00000104)

Table 1.244. 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 1.245. CMD_CREATE_INST_OPTION Field Description

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

1.2.5.2.4.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.246. 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

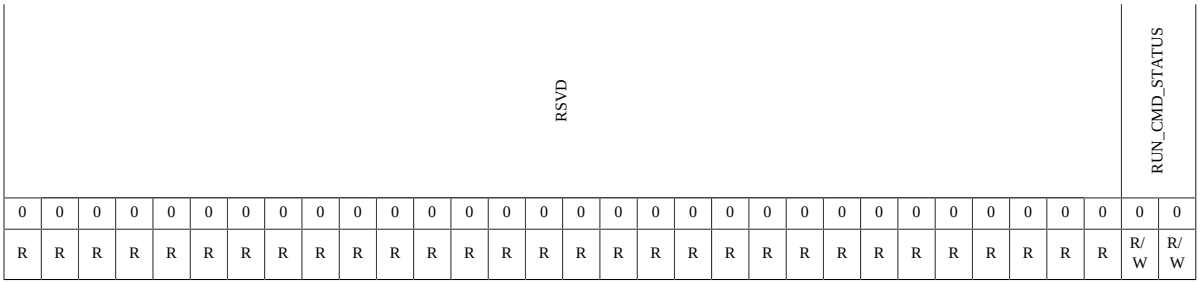


Table 1.247. RET_SUCCESS Field Description

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_WARNIG	0x0

1.2.5.2.4.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.248. RET_FAIL_REASON Bit Assignment

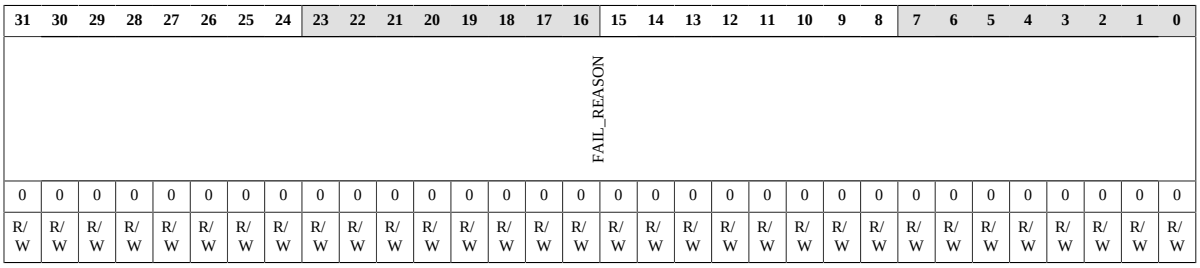


Table 1.249. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.4.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.250. CMD_INSTANCE_INFO Bit Assignment

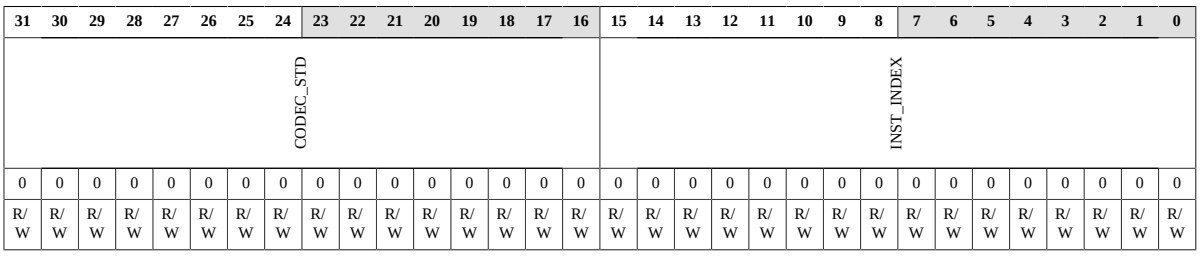


Table 1.251. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run	0x0

Bit	Name	Type	Function	Reset Value
			HEVC_DEC = 0x0 VP9_DEC = 0x16	
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.4.6. CMD_CREATE_INST_ADDR_WORK_BAS (0x00000114)

Work Buffer Base Address

Table 1.252. CMD_CREATE_INST_ADDR_WORK_BAS 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 1.253. CMD_CREATE_INST_ADDR_WORK_BAS 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 4K boundary.	0x0

1.2.5.2.4.7. CMD_CREATE_INST_WORK_SIZE (0x00000118)

Work Buffer Size

Table 1.254. 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 1.255. CMD_CREATE_INST_WORK_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	WORK_BUF_SIZE	R/W	Size of work buffer (4K boundary)	0x0

Bit	Name	Type	Function	Reset Value
			The size of work buffer should be larger than required minimum work buffer size. This address should be aligned in 4K boundary.	

1.2.5.2.4.8. CMD_CREATE_INST_BS_START_ADDR (0x0000011C)

Bitstream buffer start address

Table 1.256. CMD_CREATE_INST_BS_START_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
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 1.257. CMD_CREATE_INST_BS_START_ADDR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_START_ADDR	R/W	Start Address of Bitstream Buffer (fixed in ring buffer mode) It should be aligned in bus width (128bit/16 byte).	0x0

1.2.5.2.4.9. CMD_CREATE_INST_BS_SIZE (0x00000120)

Bitstream buffer size

Table 1.258. CMD_CREATE_INST_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
BS_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 1.259. CMD_CREATE_INST_BS_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_BUF_SIZE	R/W	Size of Bistream buffer (fixed in ring buffer mode) It should be aligned in bus width (128bit/16byte).	0x0

1.2.5.2.4.10. CMD_CREATE_INST_BS_PARAM (0x00000124)

Bitstream buffer parameters

Table 1.260. CMD_CREATE_INST_BS_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																												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
R	R	R	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

Table 1.261. CMD_CREATE_INST_BS_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	BS_ENDIAN	R/W	Endianness of bitstream buffer	0x0

1.2.5.2.4.11. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.262. 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 1.263. 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.	0x0

1.2.5.2.4.12. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.264. 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 1.265. 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 its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.4.13. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.266. 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_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.267. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.4.14. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.268. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.269. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.4.15. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.270. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.4.16. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.272. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							

0	0	0	0	0	0	0	0	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 1.273. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.5. 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.

1.2.5.2.5.1. 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 1.274. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.275. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.5.2. CMD_FLUSH_INST_OPTION (0x00000104)

Table 1.276. CMD_FLUSH_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 1.277. CMD_FLUSH_INST_OPTION Field Description

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

1.2.5.2.5.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.278. 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

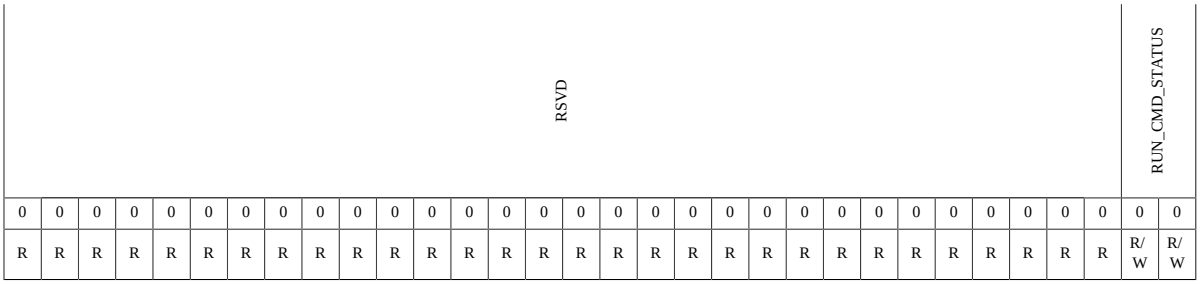


Table 1.279. RET_SUCCESS Field Description

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_WARNIG	0x0

1.2.5.2.5.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.280. RET_FAIL_REASON Bit Assignment

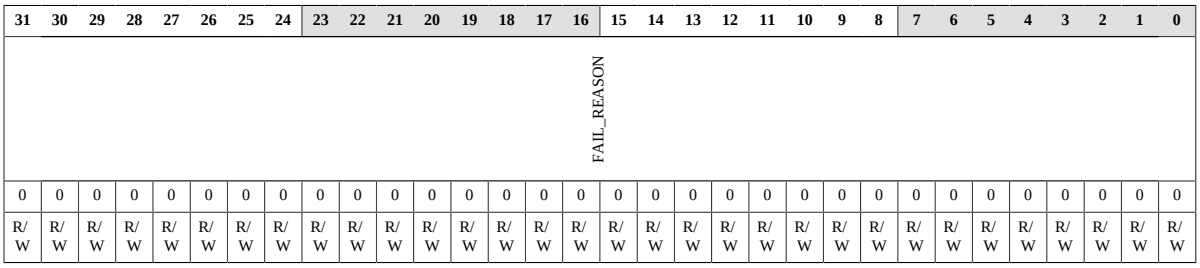


Table 1.281. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.5.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.282. CMD_INSTANCE_INFO Bit Assignment

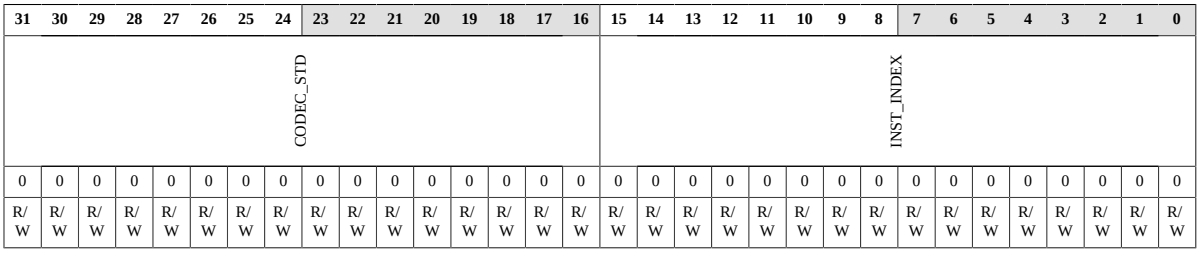


Table 1.283. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run	0x0

Bit	Name	Type	Function	Reset Value
			HEVC_DEC = 0x0 VP9_DEC = 0x16	
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.5.6. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.284. 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 1.285. 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.	0x0

1.2.5.2.5.7. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.286. 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 1.287. 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 its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.5.8. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.288. 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_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.289. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.5.9. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.290. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.5.10. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.292. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.5.11. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.294. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							

0	0	0	0	0	0	0	0	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. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.6. 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.

1.2.5.2.6.1. 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 1.296. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.6.2. CMD_DESTROY_INST_OPTION (0x00000104)

Table 1.298. CMD_DESTROY_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 1.299. CMD_DESTROY_INST_OPTION Field Description

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

1.2.5.2.6.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.300. 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
R	R	R	R	R	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 1.301. 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_WARNIG	0x0

1.2.5.2.6.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.302. 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 1.303. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.6.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.304. 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 1.305. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run	0x0

Bit	Name	Type	Function	Reset Value
			HEVC_DEC = 0x0 VP9_DEC = 0x16	
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.6.6. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.306. 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 1.307. 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.	0x0

1.2.5.2.6.7. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.308. 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 1.309. 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 its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.6.8. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.310. 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_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.311. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.6.9. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.312. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.6.10. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.314. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.315. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.6.11. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.316. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							

0	0	0	0	0	0	0	0	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 1.317. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.7. INIT_SEQ Command Parameter Registers

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

1.2.5.2.7.1. 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 **INIT_SEQ** command (**0x0040** is given to this register).

Table 1.318. 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 1.319. 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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.7.2. CMD_INIT_SEQ_OPTION (0x00000104)

Decode picture header option

Table 1.320. CMD_INIT_SEQ_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																										INIT_SEQ_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 1.321. CMD_INIT_SEQ_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:6]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[5:0]	INIT_SEQ_OPTION	R/W	0x01: INIT_SEQ 0x11: INIT_SEQ (w/ Thumbnail mode)	0x0

1.2.5.2.7.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.322. 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 1.323. 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_WARNIG	0x0

1.2.5.2.7.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.324. 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 1.325. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.7.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.326. 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

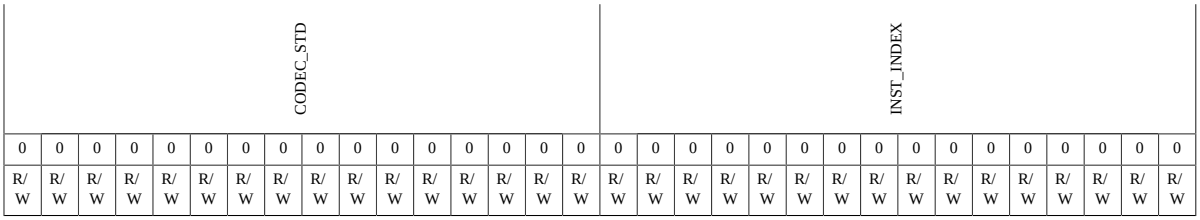


Table 1.327. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run HEVC_DEC = 0x0 VP9_DEC = 0x16	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.7.6. CMD_BS_RD_PTR (0x00000118)

Bistream Buffer Read Pointer

Table 1.328. CMD_BS_RD_PTR Bit Assignment

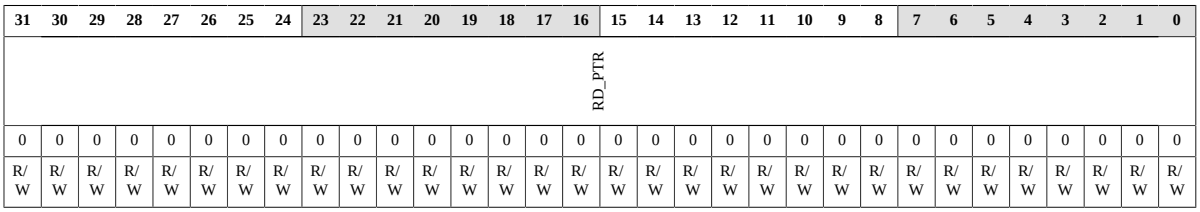


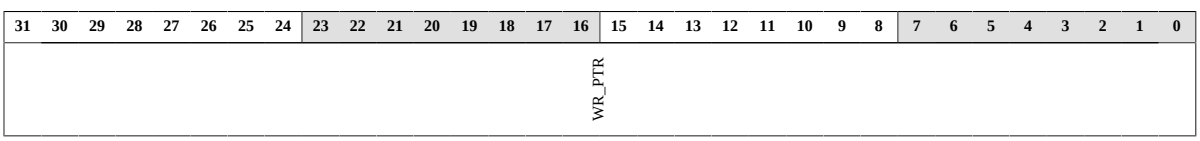
Table 1.329. CMD_BS_RD_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RD_PTR	R/W	Start address of bitstream for handling current command Host processor cannot update this register in the middle of decoding. It is only allowed to update this before decoding has started. VPU also updates this (RET_ADDR_BS_RD_PTR) with end address of a NAL, when a NAL is decoded.	0x0

1.2.5.2.7.7. CMD_BS_WR_PTR (0x0000011C)

Bistream Buffer Write Pointer

Table 1.330. CMD_BS_WR_PTR 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 1.331. CMD_BS_WR_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	WR_PTR	R/W	End address of bitstream for handling current command Host can update this register anytime. If a bitstream_empty interrupt is asserted, host processor should do either feed more bitstream and update WR_PTR or set EXPLICIT_END to complete decoding anyhow.	0x0

1.2.5.2.7.8. CMD_BS_OPTIONS (0x00000120)

Bistream buffer option

Table 1.332. CMD_BS_OPTIONS 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																														STREAM_END	EXPLICIT_END
0	0	0	0	0	0	0	0	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 1.333. CMD_BS_OPTIONS Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R/W	Reserved	0x0
[1]	STREAM_END	R/W	This field makes VPU assume Stream End when there is no stream to feed in the buffer.	0x0
[0]	EXPLICIT_END	R/W	Explicit End When this field is set to 1, VPU assumes that bitstream buffer has at least one single frame and follows the tasks below. 1. VPU decodes until the end of bitstream buffer (WR_PTR) even if the last 3 bytes are all 0. 2. VPU returns success if it has decoded a frame successfully. If bitstream is insufficient to complete decoding a frame, VPU performs what it is supposed to do with the specified task in BS_SHORTAGE_OPTION of BS_PARAM. If this flag is 0, 1. VPU decodes to the almost end of bitstream buffer (WR_PTR), but not to some bytes (less than 3). It intentionally does not consume some a few bytes, because VPU is not sure if the last bytes are the start code of next NAL or they might not fill the offset in the CABAC. 2. VPU returns success if it has decoded a frame successfully. If bitstream is insufficient to complete decoding a frame, VPU stops decoding and waits for more bitstream to be filled. Then host processor can do either feed bitstream more or set EXPLICIT_END to complete decoding anyhow.	0x0

Bit	Name	Type	Function	Reset Value
			BITSTREAM_EMPTY interrupt is asserted when EXPLITCT_END = 0 or when bitstream buffer is near empty (not real empty) for seamless decoding. **Caution** Host processor can set this register any time, but cannot clear during command processing.	

1.2.5.2.7.9. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.334. 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 1.335. 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.	0x0

1.2.5.2.7.10. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.336. 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 1.337. 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 its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.7.11. RET_QUE_STATUS (0x000001EC)

Table 1.338. RET_QUE_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 1.339. RET_QUE_STATUS Field Description

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

1.2.5.2.7.12. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.340. 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_INSTANCE_INFO							

0	0	0	0	0	0	0	0	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	R/W

Table 1.341. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.7.13. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.342. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.7.14. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.344. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.7.15. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.346. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.347. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.8. SET_FB Command Parameter Registers

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

1.2.5.2.8.1. 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 **SET_FB** command (**0x0080** is given to this register).

Table 1.348. 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 1.349. 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 : DEC_PIC 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.8.2. CMD_SET_FB_OPTION (0x00000104)

Table 1.350. 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						FBC_MODE_DISALBE_FLAG						RSVD																END_FLAG	INIT_FLAG	SET_FB_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	R	R	R	R	R	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW

Table 1.351. CMD_SET_FB_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31:26]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[25:20]	FBC_MODE_DISALBE_FLAG	RW	fbc_mode disable flag. Default value is 0x0C for the best compression and throughtput performance.	0x0
[19:5]	RSVD	R	Reserved	0x0
[4]	END_FLAG	RW	end_flag for the last SET_FB command If SET_FB_MODE = 1, END_FLAG is refered to as 1.	0x0
[3]	INIT_FLAG	RW	init_flag for the first SET_FB command If SET_FB_MODE = 1, INIT_FLAG is refered to as 1.	0x0
[2:0]	SET_FB_MODE	RW	If SET_FB_MODE is equal to 1, Host replaces single BWB frame buffer and FBC frame buffer ColMv buffer as indicated by MVCOL_REPLACE_IDX, LIN_REPLACE_IDX and FBC_REPLACE_IDX. If SET_FB_MODE is equal to 0, Host provides frame buffer bases from FB_NUM_START to FB_NUM_END with more than one SET_FB command. INIT_FLAG shoud be 1 at the first SET_FB command and END_FLAG should be 1 at the last SET_FB command.	0x0

1.2.5.2.8.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.352. 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 1.353. 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_WARNIG	0x0

1.2.5.2.8.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.354. 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

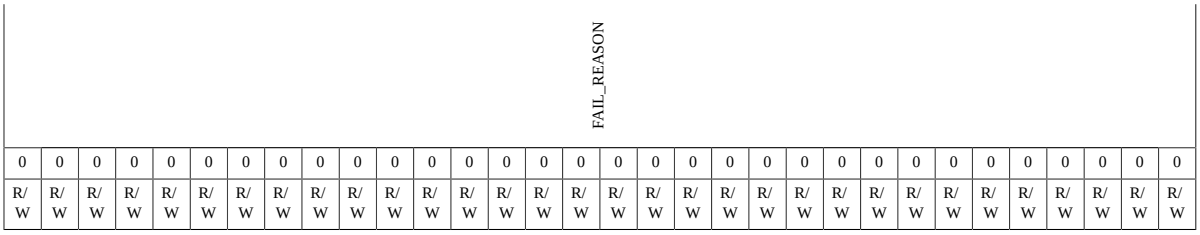


Table 1.355. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.8.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.356. CMD_INSTANCE_INFO Bit Assignment

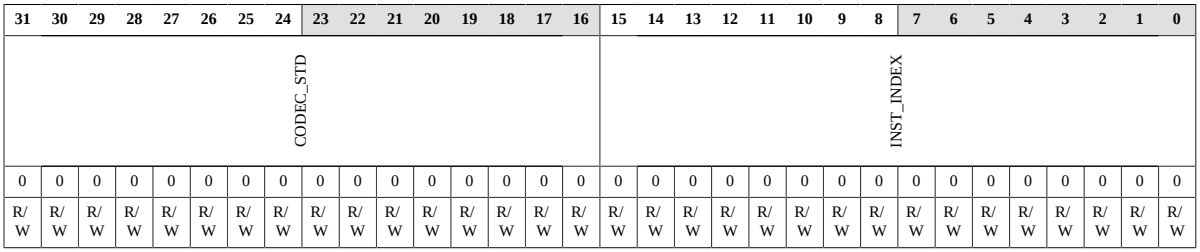


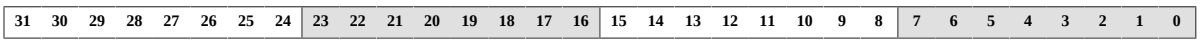
Table 1.357. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run HEVC_DEC = 0x0 VP9_DEC = 0x16	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.8.6. CMD_SET_FB_COMMON_PIC_INFO (0x00000118)

Dpb Information
This fields are valid only if SET_FB_MODE = 0.

Table 1.358. CMD_SET_FB_COMMON_PIC_INFO Bit Assignment



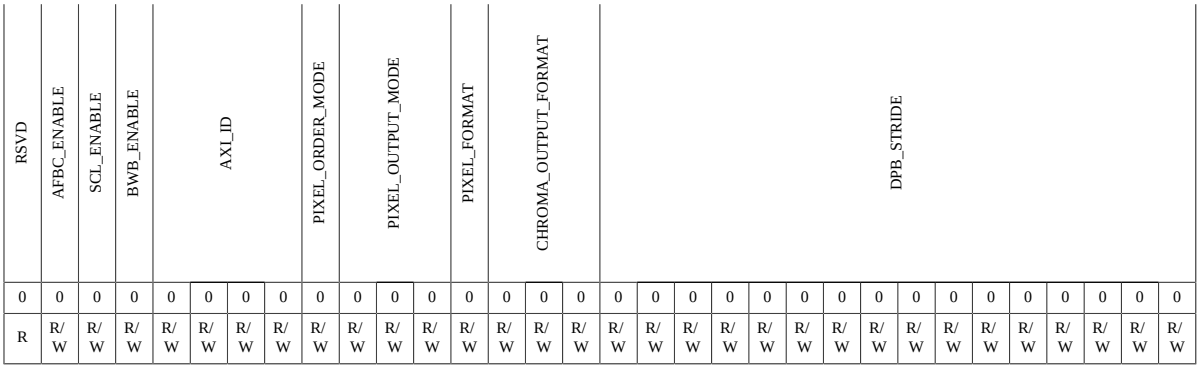


Table 1.359. CMD_SET_FB_COMMON_PIC_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31]	RSVD	R	Reserved	0x0
[30]	AFBC_ENABLE	R/W	This field enables VPU to generate AFBC(Arm Frame Buffer Compression) format of decoded frames. 0: BWB format 1: AFBC format **Note** It is valid only when AFBC block is used.	0x0
[29]	SCL_ENABLE	R/W	This field indicates whether output scaling is enabled or not. 0: Scaling disable 1: Scaling enable This field is valid only if BWB_ENABLE = 1	0x0
[28]	BWB_ENABLE	R/W	This field indicates whether target frame buffers are used for uncompressed linear frame. 0: Target frame buffers are compressed frames. 1: Target frame buffers are uncompressed linear frames.	0x0
[27:24]	AXI_ID	R/W	AXI_ID to distinguish guest OS For virtualization only. Set this value in highest bit order.	0x0
[23]	PIXEL_ORDER_MODE	R/W	Pixel ordering in a bus word 0: decreasing ordering - pixel position is decreasing as byte address in a bus is increaing. 1: incrsing ordering - pixel position is increasing as byte address in a bus is increaing. **Note** Pixel position is always increasing as word address is increasing. This field is valid only if BWB_ENABLE = 1	0x0
[22:20]	PIXEL_OUTPUT_MODE	R/W	[22] 0 : MSB justified pixel 1 : LSB justified pixel [21:20] 0 : BWB output mode 0 (10bit/pixel -> 8bit/pixel) 1 : BWB output mode 1 (16bit/pixel) 2 : BWB output mode 2 (32bit/pixel) PIXEL_OUTPUT_MODE cannot be 2 if Chroma Output Format (COMMON_PIC_INFO[18:16]) is set as packed mode (4~7)	0x0

Bit	Name	Type	Function	Reset Value
			It is valid only if BWB_ENABLE = 1 or SCALER_ENABLE = 1.	
[19]	PIXEL_FORMAT	R/W	0 : YCbCr420 1 : YCbCr422 This field is valid only if BWB_ENABLE = 1 and SCL_ENABLE = 1. The default value is 0.	0x0
[18:16]	CHROMA_OUTPUT_FORMAT	R/W	BWB Chroma format 0 : Planar mode - YV12 when YCbCr420, YV16 when YCbCr422 1 : Semi-planar mode, Cb/Cr interleaved - NV12 when YCbCr420, NV16 when YCbCr422 3 : Semi-planar mode, Cb/Cr interleaved - NV21 when YCbCr420, NV61 when YCbCr422 This field is valid only if BWB_ENABLE = 1.	0x0
[15:0]	DPB_STRIDE	R/W	Stride for the storing alignment	0x0

1.2.5.2.8.7. CMD_SET_FB_STRIDE (0x00000118)

replace Dpb Information
This fields are valid only if SET_FB_MODE = 1.

Table 1.360. CMD_SET_FB_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
AFBC_ENABLE																AFBC_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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_SET_FB_STRIDE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	AFBC_ENABLE	R/W	Linear frame buffer stride	0x0
[15:0]	AFBC_ENABLE	R/W	fbc frame buffer stride	0x0

1.2.5.2.8.8. CMD_SET_FB_PIC_SIZE (0x0000011C)

Decoded Picture Size

Table 1.362. 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
DPB_WIDTH																DPB_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 1.363. CMD_SET_FB_PIC_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	DPB_WIDTH	R/W	Dpb Picture Width	0x0
[15:0]	DPB_HEIGHT	R/W	Dpb Picture Height	0x0

1.2.5.2.8.9. CMD_SET_FB_SET_FB_NUM (0x00000120)

Option of set frame buffer

Table 1.364. CMD_SET_FB_SET_FB_NUM 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_NUM_START					RSVD			FB_NUM_END				
0	0	0	0	0	0	0	0	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 1.365. CMD_SET_FB_SET_FB_NUM Field Description

Bit	Name	Type	Function	Reset Value
[31:13]	RSVD	R	Reserved	0x0
[12:8]	FB_NUM_START	R/W	The number of start frame buffer to set using SET_FRAME_BUFFER comamnd. This field is valid only if SET_FB_MODE = 0.	0x0
[7:5]	RSVD	R	Reserved	0x0
[4:0]	FB_NUM_END	R/W	The number of end frame buffer to set using SET_FRAME_BUFFER comamnd. This field is valid only if SET_FB_MODE = 0.	0x0

1.2.5.2.8.10. SET_FB_INDICE (0x00000120)

This field is valid only if SET_FB_MODE = 1.

Table 1.366. SET_FB_INDICE 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								MVCOL_REPLACE_IDX								LIN_REPLACE_IDX								FBC_REPLACE_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.367. SET_FB_INDICE Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[23:16]	MVCOL_REPLACE_IDX	R/W	The Col-mv buffer index to be replaced with CMD_SET_FB_ADDR_MV_COL	0x0
[15:8]	LIN_REPLACE_IDX	R/W	The linear frame buffer index to replaced with CMD_SET_FB_ADDR_LUMA_BASE	0x0
[7:0]	FBC_REPLACE_IDX	R/W	FBC frame buffer index to be replaced with CMD_SET_FB_ADDR_FBC_Y_BASE	0x0

1.2.5.2.8.11. CMD_SET_FB_SCL_OUT_SIZE (0x00000124)

Scaler Output Picture Size

Table 1.368. CMD_SET_FB_SCL_OUT_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
SCL_WIDTH																SCL_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 1.369. CMD_SET_FB_SCL_OUT_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	SCL_WIDTH	R/W	SCL_OUT_X_SIZE The output horizontal size of scaler which should be the multiple of 8. It should be less than or equal to the width of decoding frame. But, it should be greater than or equal to the 1/8 of the width of decoding frame. **Note** This field is valid only if SCALER_ENABLE = 1	0x0
[15:0]	SCL_HEIGHT	R/W	SCL_OUT_Y_SIZE The output vertical size of scaler which should be the multiple of 8. It should be less than or equal to the height of decoding frame. But, it should be greater than or equal to the 1/8 of the height of decoding frame. **Note** This field is valid only if SCALER_ENABLE = 1	0x0

1.2.5.2.8.12. CMD_SET_FB_ADDR_LUMA_BASE0 (0x00000134)

Luma base of index0

Note CMD_SET_FB_ADDR_LUMA_BASE0(0x134) to CMD_SET_FB_ADDR_MV_COL7(0x1D0) are valid only if SET_FB_MODE is 0.

Table 1.370. 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 1.371. 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 or AFBC is used. If FBC is used, this is compressed Luma base address of DPB index 0. If AFBC is used, this is AFBC-compressed luma/chroma base address of DPB index 0.	0x0

1.2.5.2.8.13. CMD_SET_FB_ADDR_LUMA_BASE (0x00000134)

Luma base of index LIN_REPLACE_IDX
NOTE . CMD_SET_FB_ADDR_LUMA_BASE to CMD_SET_FB_ADDR_FBC_C_OFFSET are valid only if SET_FB_MODE = 1.

Table 1.372. CMD_SET_FB_ADDR_LUMA_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
LUMA_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 1.373. CMD_SET_FB_ADDR_LUMA_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE	R/W	Luma base address of DPB. If AFBC is used, this is AFBC-compressed luma/chroma base address of DPB. If LIN_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.14. CMD_SET_FB_ADDR_CB_BASE0 (0x00000138)

Cb base of index0

Table 1.374. 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 1.375. 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

1.2.5.2.8.15. CMD_SET_FB_ADDR_CB_BASE (0x00000138)

Cb base of index LIN_REPLACE_IDX

Table 1.376. CMD_SET_FB_ADDR_CB_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
CB_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 1.377. CMD_SET_FB_ADDR_CB_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE	R/W	Cb base address of DPB. If LIN_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.16. CMD_SET_FB_ADDR_CR_BASE0 (0x0000013C)

Cr base of index0

Table 1.378. 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 1.379. 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

1.2.5.2.8.17. CMD_SET_FB_ADDR_FBC_Y_OFFSET0 (0x0000013C)

FBC Luma offset base of index0

Table 1.380. 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 1.381. 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

1.2.5.2.8.18. CMD_SET_FB_ADDR_CR_BASE (0x0000013C)

Cr base of index LIN_REPLACE_IDX

Table 1.382. CMD_SET_FB_ADDR_CR_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
CR_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 1.383. CMD_SET_FB_ADDR_CR_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CR_BASE	R/W	Cr base address of DPB. If LIN_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.19. CMD_SET_FB_ADDR_FBC_C_OFFSET0 (0x00000140)

FBC Chroma offset base of index0

Table 1.384. CMD_SET_FB_ADDR_FBC_C_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_CHROMA_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 1.385. 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

1.2.5.2.8.20. CMD_SET_FB_ADDR_MV_COL (0x00000140)

Colocated mv buffer base of index MVCOL_REPLACE_IDX

Table 1.386. CMD_SET_FB_ADDR_MV_COL 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 1.387. CMD_SET_FB_ADDR_MV_COL Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE0	R/W	Base address of colocated motion vecotors buffer. If MVCOL_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.21. CMD_SET_FB_ADDR_LUMA_BASE1 (0x00000144)

Luma base of index1

Table 1.388. CMD_SET_FB_ADDR_LUMA_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
LUMA_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 1.389. 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 or AFBC is used. If FBC is used, this is compressed Luma base address of DPB index 1. If AFBC is used, this is AFBC-compressed luma/chroma base address of DPB index 1.	0x0

1.2.5.2.8.22. CMD_SET_FB_ADDR_FBC_Y_BASE (0x00000144)

Luma base of index FBC_REPLACE_IDX

Table 1.390. CMD_SET_FB_ADDR_FBC_Y_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
LUMA_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 1.391. CMD_SET_FB_ADDR_FBC_Y_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	LUMA_BASE	R/W	FBC compressed Luma base address. If FBC_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.23. CMD_SET_FB_ADDR_CB_BASE1 (0x00000148)

Cb base of index1

Table 1.392. 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 1.393. 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

1.2.5.2.8.24. CMD_SET_FB_ADDR_FBC_C_BASE (0x00000148)

Cb base of index FBC_REPLACE_IDX

Table 1.394. CMD_SET_FB_ADDR_FBC_C_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
CB_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 1.395. CMD_SET_FB_ADDR_FBC_C_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CB_BASE	R/W	FBC compressed Cb and Cr base address. If FBC_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.25. CMD_SET_FB_ADDR_CR_BASE1 (0x0000014C)

Cr base of index1

Table 1.396. 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 1.397. 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

1.2.5.2.8.26. CMD_SET_FB_ADDR_FBC_Y_OFFSET1 (0x0000014C)

FBC luma offset base of index1

Table 1.398. 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 1.399. 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

1.2.5.2.8.27. CMD_SET_FB_ADDR_FBC_Y_OFFSET (0x0000014C)

FBC Luma offset base of index FBC_REPLACE_IDX

Table 1.400. CMD_SET_FB_ADDR_FBC_Y_OFFSET 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_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 1.401. CMD_SET_FB_ADDR_FBC_Y_OFFSET Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_LUMA_OFFSET_BASE	R/W	FBC Compressed Luma offset table base address. If FBC_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.28. CMD_SET_FB_ADDR_FBC_C_OFFSET1 (0x00000150)

FBC chroma offset base of index1

Table 1.402. CMD_SET_FB_ADDR_FBC_C_OFFSET1 Bit Assignment

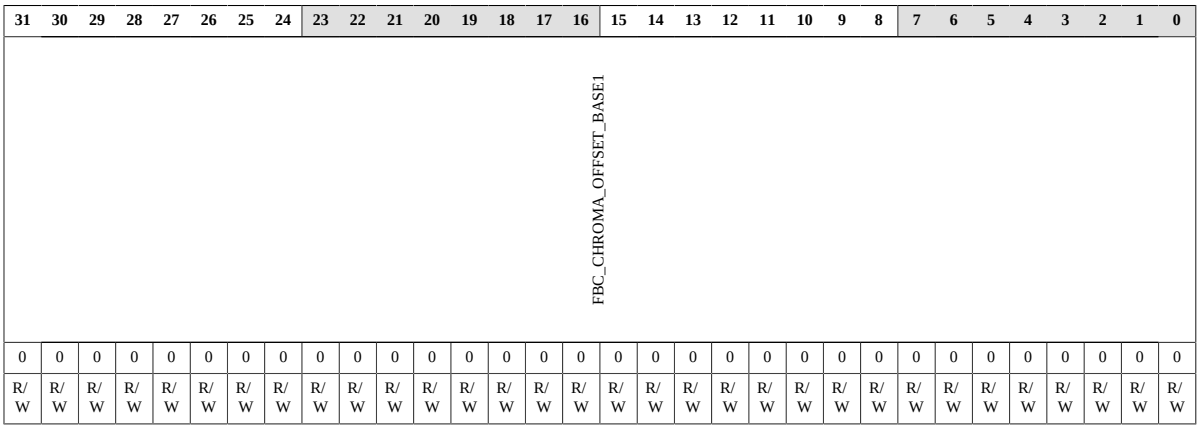


Table 1.403. CMD_SET_FB_ADDR_FBC_C_OFFSET1 Field Description

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

1.2.5.2.8.29. CMD_SET_FB_ADDR_FBC_C_OFFSET (0x00000150)

FBC Chroma offset base of index FBC_REPLACE_IDX

Table 1.404. CMD_SET_FB_ADDR_FBC_C_OFFSET Bit Assignment

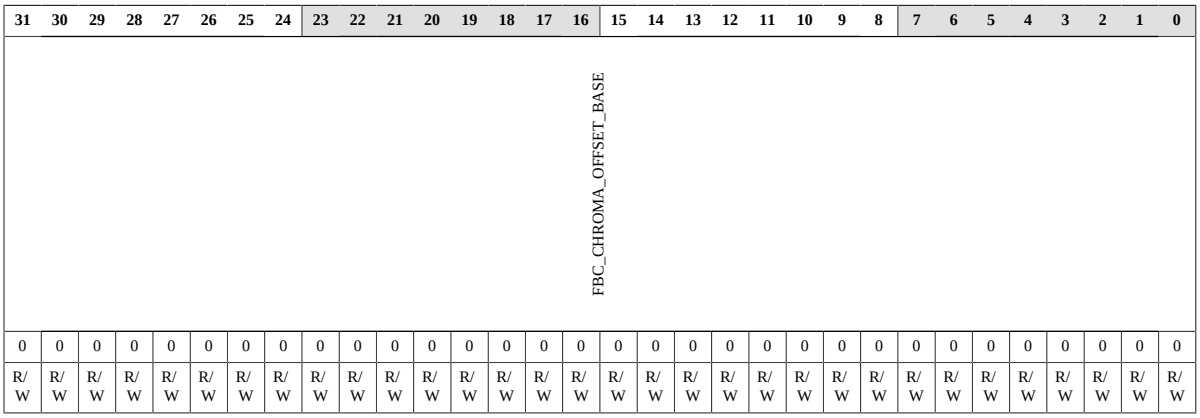


Table 1.405. CMD_SET_FB_ADDR_FBC_C_OFFSET Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FBC_CHROMA_OFFSET_BASE	R/W	FBC Compressed chroma offset base address. If FBC_REPLACE_IDX < 0, it is ignored.	0x0

1.2.5.2.8.30. CMD_SET_FB_ADDR_LUMA_BASE2 (0x00000154)

Luma base of index2

Table 1.406. 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 1.407. 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 or AFBC is used. If FBC is used, this is compressed Luma base address of DPB index 2. If AFBC is used, this is AFBC-compressed luma/chroma base address of DPB index 2.	0x0

1.2.5.2.8.31. CMD_SET_FB_ADDR_CB_BASE2 (0x00000158)

Cb base of index2

Table 1.408. 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 1.409. 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

1.2.5.2.8.32. CMD_SET_FB_ADDR_CR_BASE2 (0x0000015C)

Cr base of index2

Table 1.410. 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 1.411. 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

1.2.5.2.8.33. CMD_SET_FB_ADDR_FBC_C_OFFSET2 (0x00000160)

FBC chroma offset base of index2

Table 1.412. CMD_SET_FB_ADDR_FBC_C_OFFSET2 Bit Assignment

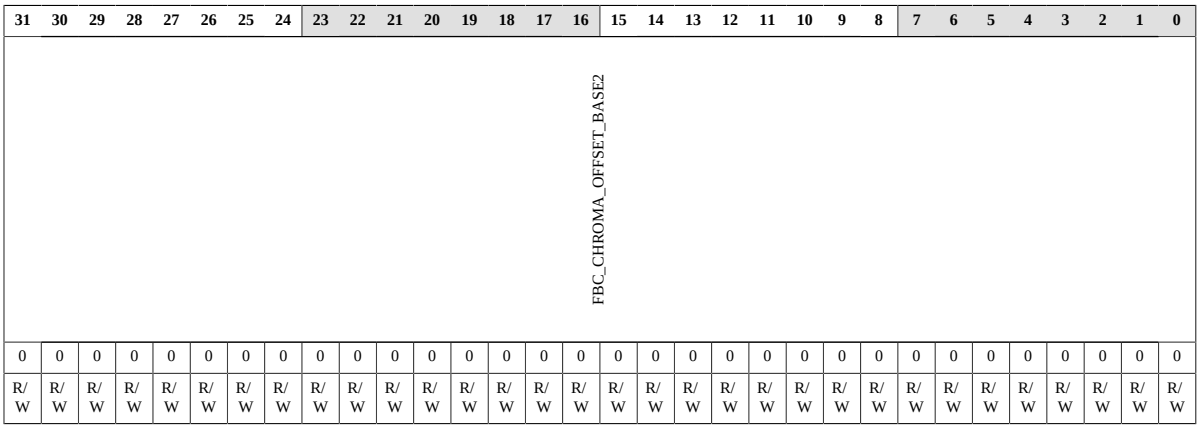


Table 1.413. CMD_SET_FB_ADDR_FBC_C_OFFSET2 Field Description

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

1.2.5.2.8.34. CMD_SET_FB_ADDR_LUMA_BASE3 (0x00000164)

Luma base of index3

Table 1.414. CMD_SET_FB_ADDR_LUMA_BASE3 Bit Assignment

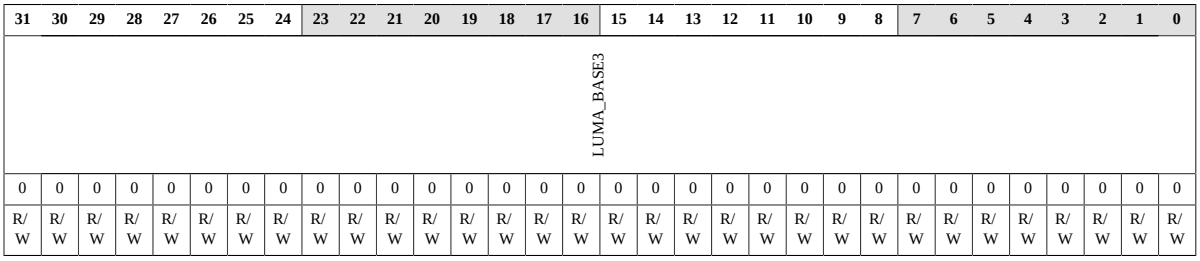


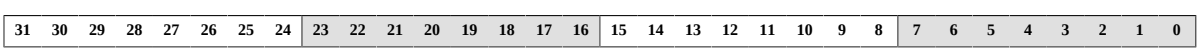
Table 1.415. CMD_SET_FB_ADDR_LUMA_BASE3 Field Description

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

1.2.5.2.8.35. CMD_SET_FB_ADDR_CB_BASE3 (0x00000168)

Cb base of index3

Table 1.416. CMD_SET_FB_ADDR_CB_BASE3 Bit Assignment



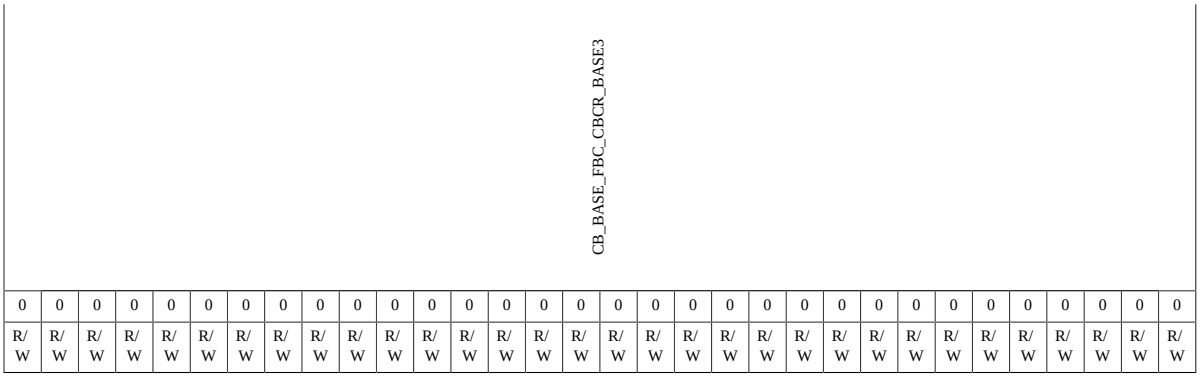


Table 1.417. CMD_SET_FB_ADDR_CB_BASE3 Field Description

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

1.2.5.2.8.36. CMD_SET_FB_ADDR_CR_BASE3 (0x0000016C)

Cr base of index3

Table 1.418. CMD_SET_FB_ADDR_CR_BASE3 Bit Assignment

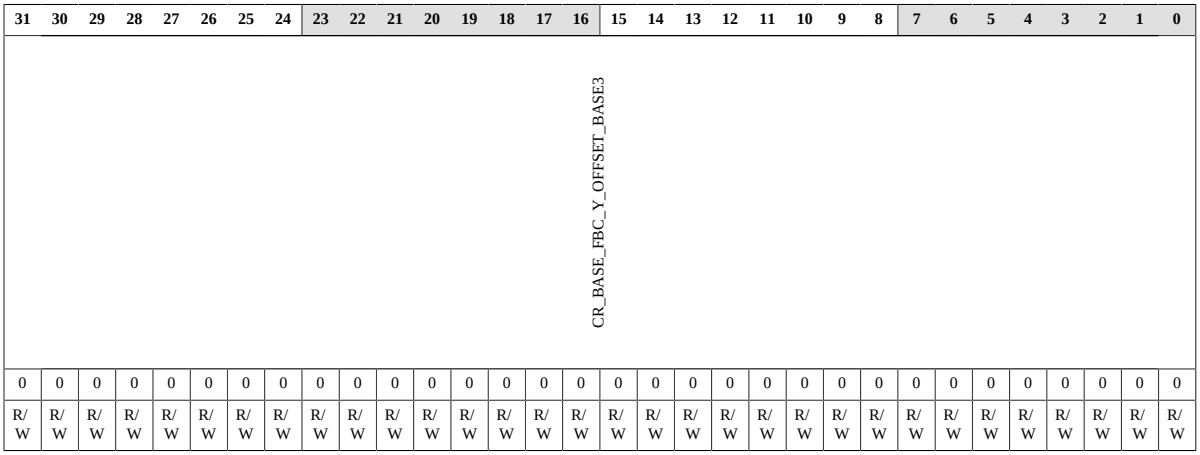


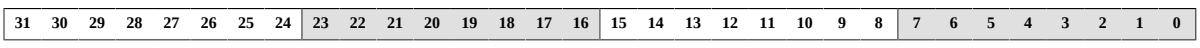
Table 1.419. CMD_SET_FB_ADDR_CR_BASE3 Field Description

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

1.2.5.2.8.37. CMD_SET_FB_ADDR_FBC_Y_OFFSET3 (0x0000016C)

FBC luma offset base of index3

Table 1.420. CMD_SET_FB_ADDR_FBC_Y_OFFSET3 Bit Assignment



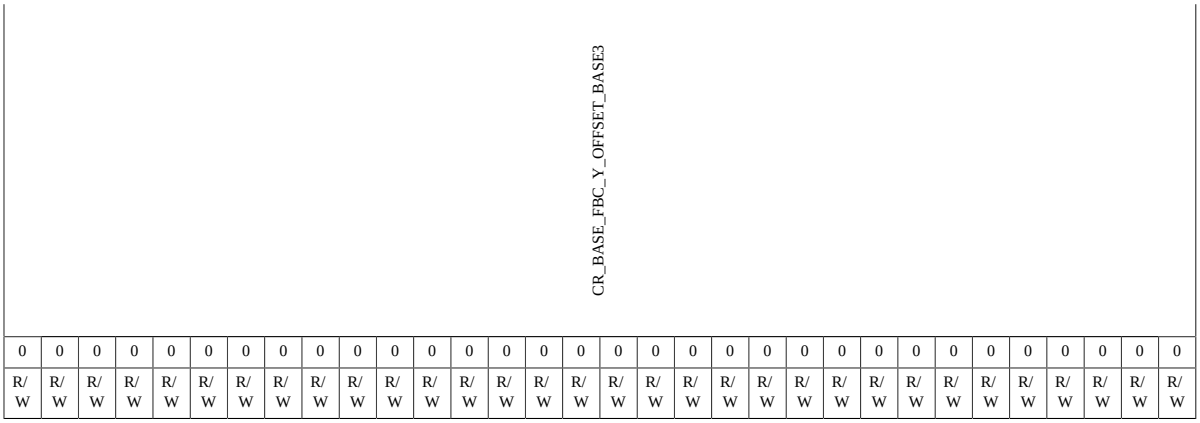


Table 1.421. CMD_SET_FB_ADDR_FBC_Y_OFFSET3 Field Description

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

1.2.5.2.8.38. CMD_SET_FB_ADDR_FBC_C_OFFSET3 (0x00000170)

FBC Chroma offset base of index3

Table 1.422. CMD_SET_FB_ADDR_FBC_C_OFFSET3 Bit Assignment

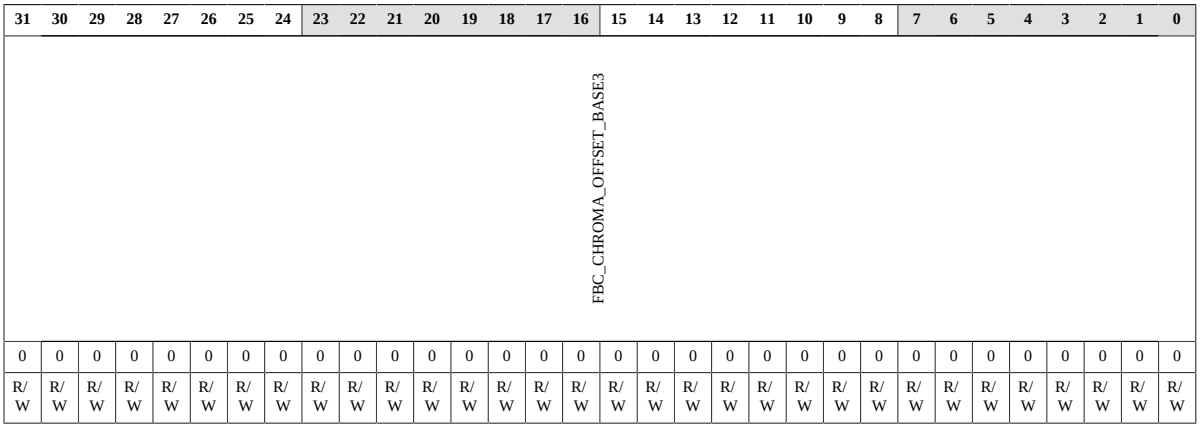


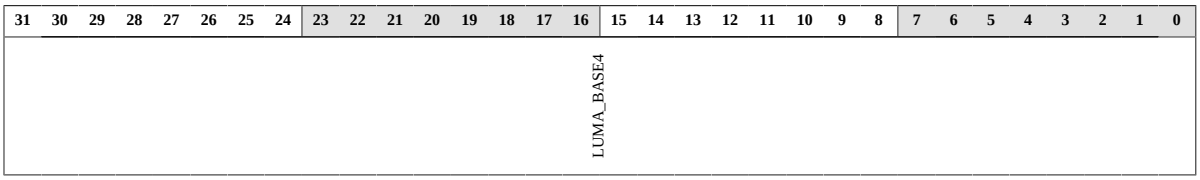
Table 1.423. CMD_SET_FB_ADDR_FBC_C_OFFSET3 Field Description

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

1.2.5.2.8.39. CMD_SET_FB_ADDR_LUMA_BASE4 (0x00000174)

Luma base of index4

Table 1.424. CMD_SET_FB_ADDR_LUMA_BASE4 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 1.425. 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 or AFBC is used. If FBC is used, this is compressed Luma base address of DPB index 4. If AFBC is used, this is AFBC-compressed luma/chroma base address of DPB index 4.	0x0

1.2.5.2.8.40. CMD_SET_FB_ADDR_CB_BASE4 (0x00000178)

Cb base of index4

Table 1.426. 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
CB_BASE_FBC_CBCR_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 1.427. 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

1.2.5.2.8.41. CMD_SET_FB_ADDR_CR_BASE4 (0x0000017C)

Cr base of index4

Table 1.428. CMD_SET_FB_ADDR_CR_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
CR_BASE_FBC_Y_OFFSET_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 1.429. 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

1.2.5.2.8.42. CMD_SET_FB_ADDR_FBC_Y_OFFSET4 (0x0000017C)

Cr base of index4

Table 1.430. CMD_SET_FB_ADDR_FBC_Y_OFFSET4 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_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 1.431. 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

1.2.5.2.8.43. CMD_SET_FB_ADDR_FBC_C_OFFSET4 (0x00000180)

FBC Chroma offset base of index4

Table 1.432. CMD_SET_FB_ADDR_FBC_C_OFFSET4 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_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 1.433. 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

1.2.5.2.8.44. CMD_SET_FB_ADDR_LUMA_BASE5 (0x00000184)

Luma base of index5

Table 1.434. CMD_SET_FB_ADDR_LUMA_BASE5 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_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 1.435. 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 or AFBC is used. If FBC is used, this is compressed Luma base address of DPB index 5. If AFBC is used, this is AFBC-compressed luma/chroma base address of DPB index 5.	0x0

1.2.5.2.8.45. CMD_SET_FB_ADDR_CB_BASE5 (0x00000188)

Cb base of index5

Table 1.436. CMD_SET_FB_ADDR_CB_BASE5 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_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 1.437. 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

1.2.5.2.8.46. CMD_SET_FB_ADDR_CR_BASE5 (0x0000018C)

Cr base of index5

Table 1.438. CMD_SET_FB_ADDR_CR_BASE5 Bit Assignment

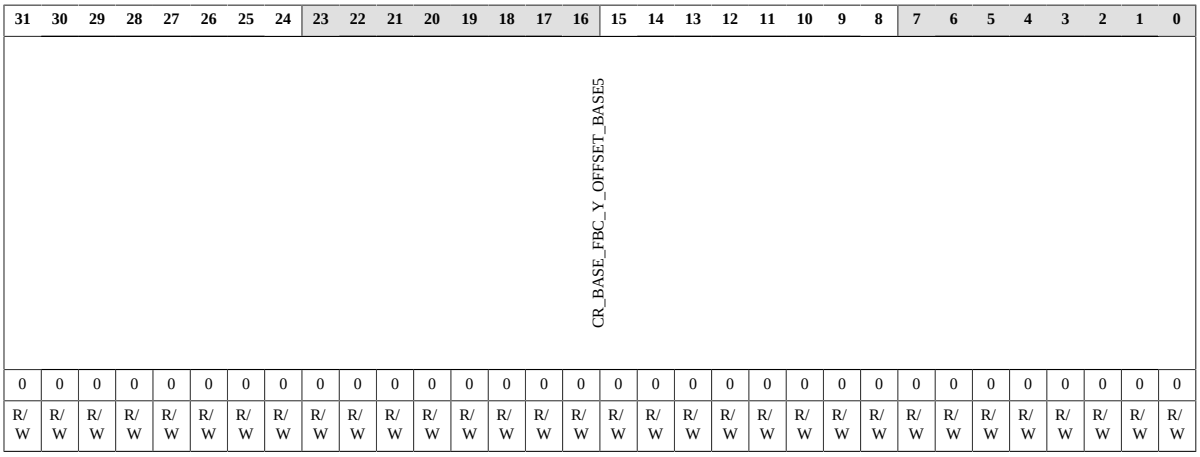


Table 1.439. CMD_SET_FB_ADDR_CR_BASE5 Field Description

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

1.2.5.2.8.47. CMD_SET_FB_ADDR_FBC_Y_OFFSET5 (0x0000018C)

FBC luma offset base of index5

Table 1.440. CMD_SET_FB_ADDR_FBC_Y_OFFSET5 Bit Assignment

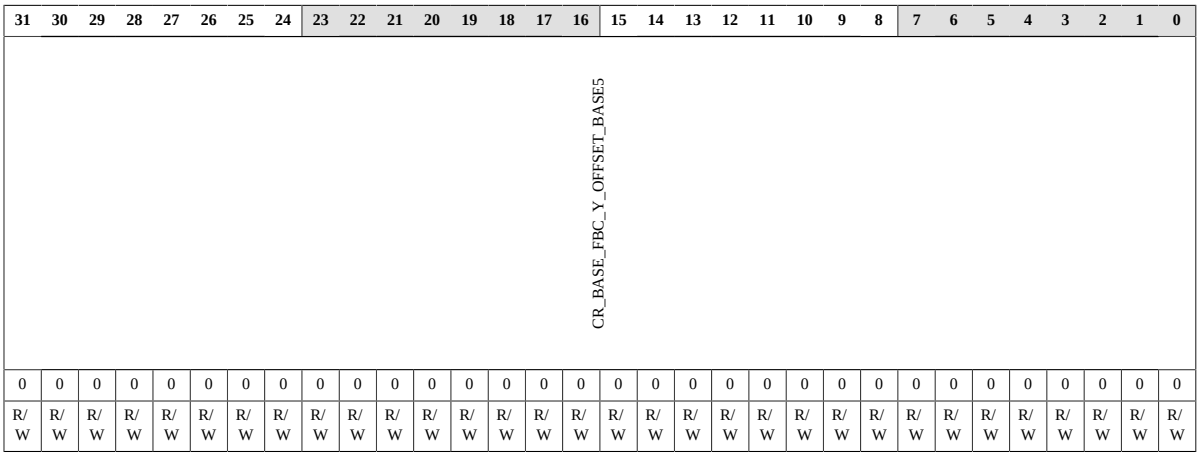


Table 1.441. CMD_SET_FB_ADDR_FBC_Y_OFFSET5 Field Description

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

1.2.5.2.8.48. CMD_SET_FB_ADDR_FBC_C_OFFSET5 (0x00000190)

FBC chroma offset base of index5

Table 1.442. CMD_SET_FB_ADDR_FBC_C_OFFSET5 Bit Assignment

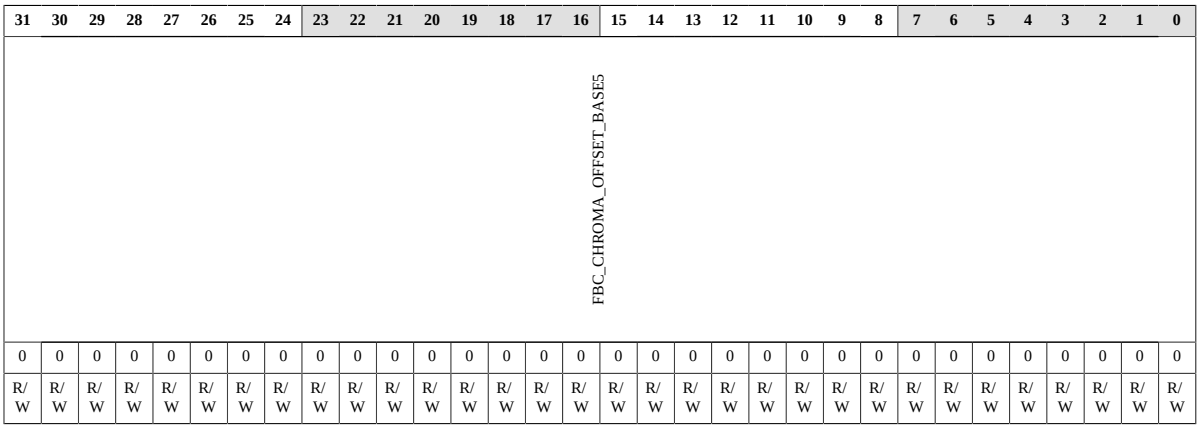


Table 1.443. CMD_SET_FB_ADDR_FBC_C_OFFSET5 Field Description

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

1.2.5.2.8.49. CMD_SET_FB_ADDR_LUMA_BASE6 (0x00000194)

Luma base of index6

Table 1.444. CMD_SET_FB_ADDR_LUMA_BASE6 Bit Assignment

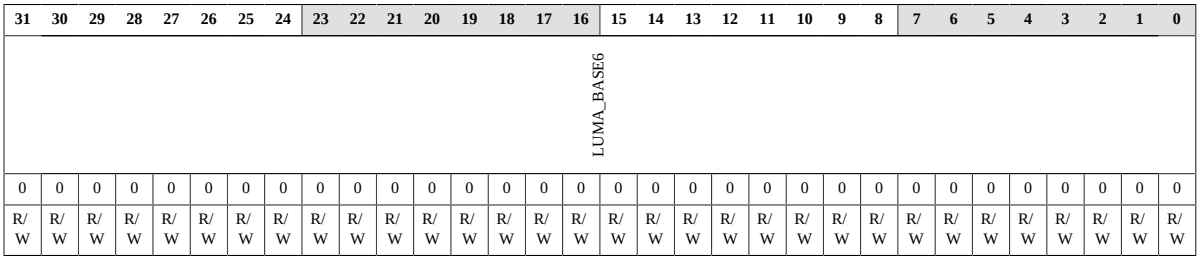


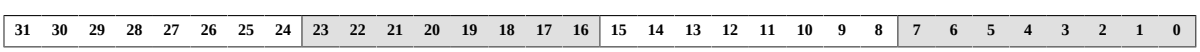
Table 1.445. CMD_SET_FB_ADDR_LUMA_BASE6 Field Description

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

1.2.5.2.8.50. CMD_SET_FB_ADDR_CB_BASE6 (0x00000198)

Cb base of index6

Table 1.446. CMD_SET_FB_ADDR_CB_BASE6 Bit Assignment



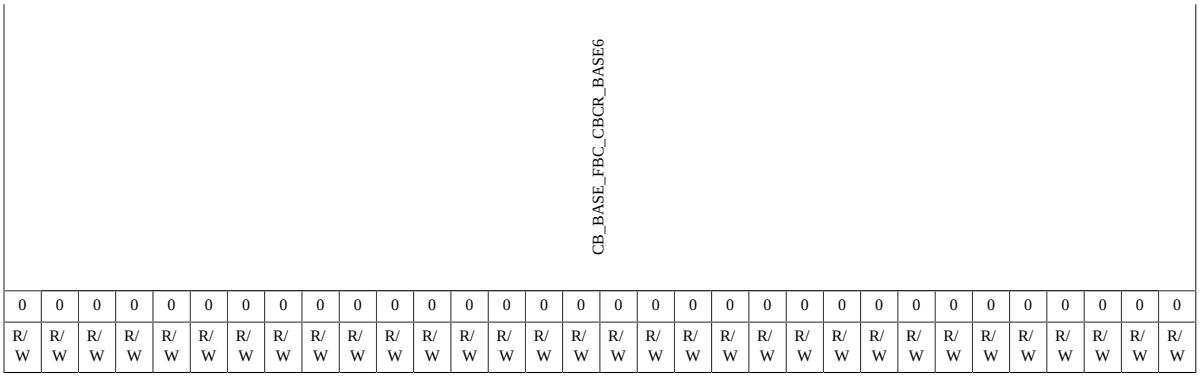


Table 1.447. CMD_SET_FB_ADDR_CB_BASE6 Field Description

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

1.2.5.2.8.51. CMD_SET_FB_ADDR_CR_BASE6 (0x0000019C)

Cr base of index6

Table 1.448. CMD_SET_FB_ADDR_CR_BASE6 Bit Assignment

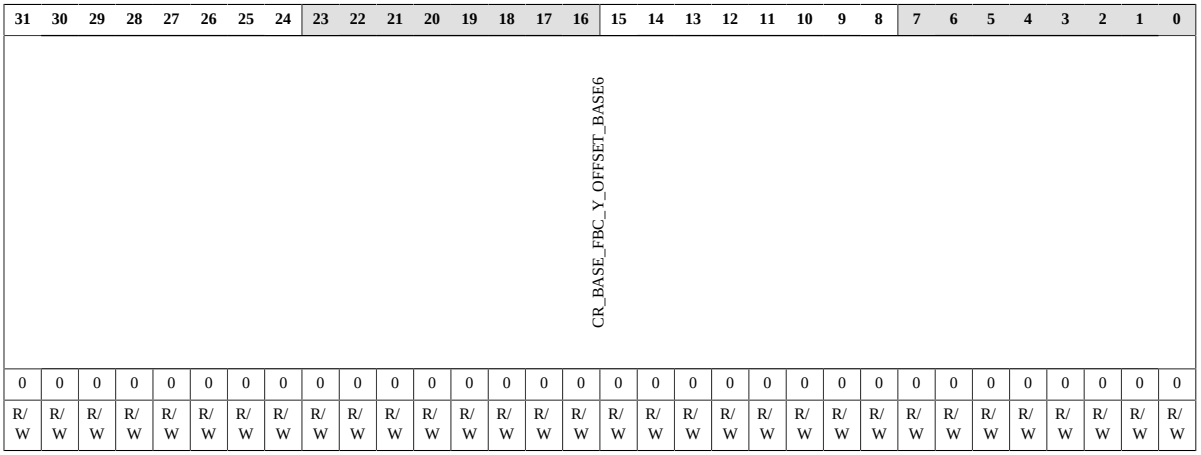


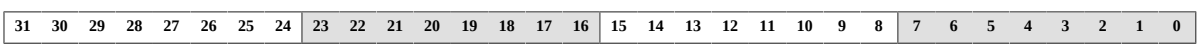
Table 1.449. CMD_SET_FB_ADDR_CR_BASE6 Field Description

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

1.2.5.2.8.52. CMD_SET_FB_ADDR_FBC_Y_OFFSET6 (0x0000019C)

FBC luma offset base of index6

Table 1.450. CMD_SET_FB_ADDR_FBC_Y_OFFSET6 Bit Assignment



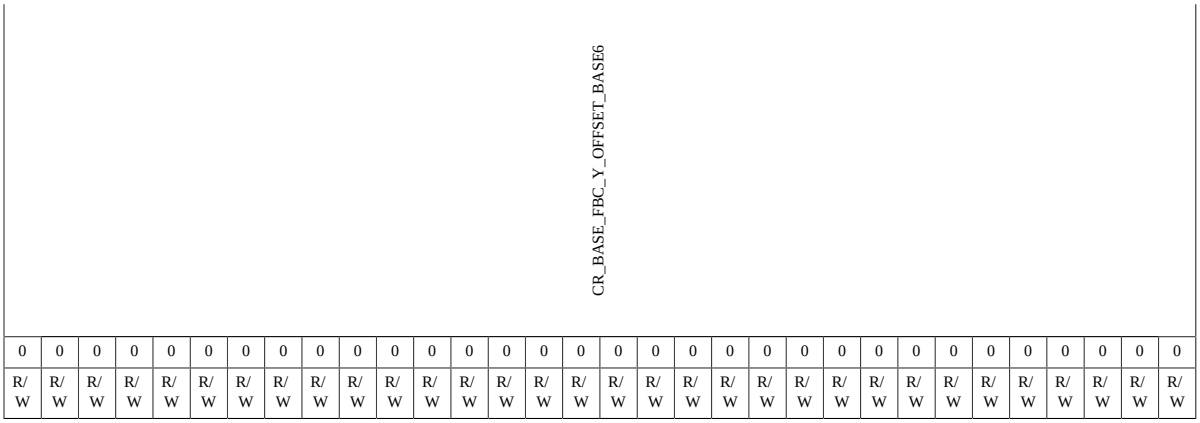


Table 1.451. CMD_SET_FB_ADDR_FBC_Y_OFFSET6 Field Description

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

1.2.5.2.8.53. CMD_SET_FB_ADDR_FBC_C_OFFSET6 (0x000001A0)

FBC chroma offset base of index6

Table 1.452. CMD_SET_FB_ADDR_FBC_C_OFFSET6 Bit Assignment

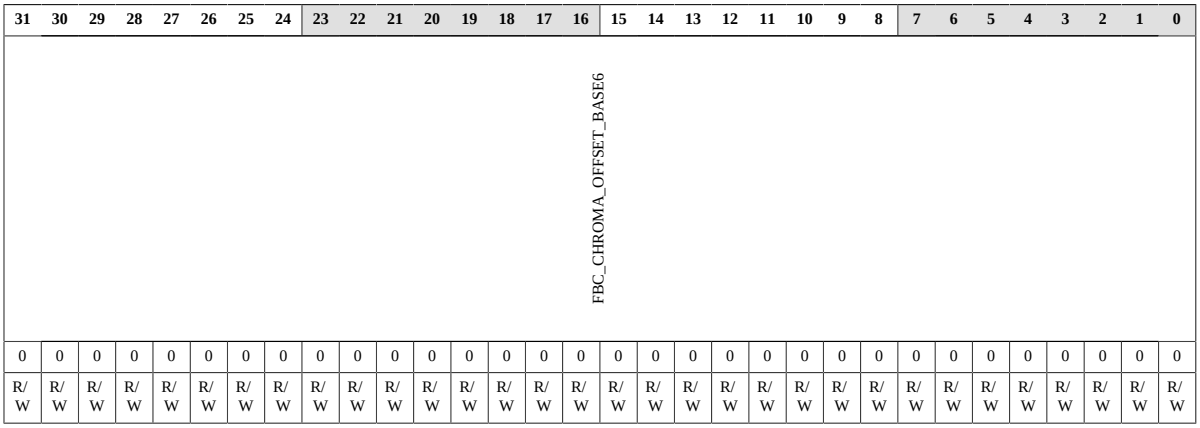


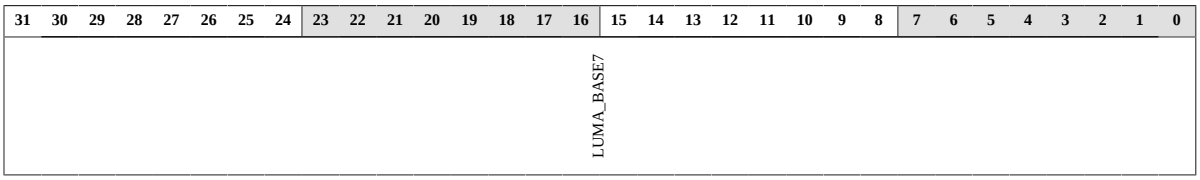
Table 1.453. CMD_SET_FB_ADDR_FBC_C_OFFSET6 Field Description

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

1.2.5.2.8.54. CMD_SET_FB_ADDR_LUMA_BASE7 (0x000001A4)

Luma base of index7

Table 1.454. CMD_SET_FB_ADDR_LUMA_BASE7 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 1.455. 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 or AFBC is used. If FBC is used, this is compressed Luma base address of DPB index 7. If AFBC is used, this is AFBC-compressed luma/chroma base address of DPB index 7.	0x0

1.2.5.2.8.55. CMD_SET_FB_ADDR_CB_BASE7 (0x000001A8)

Cb base of index7

Table 1.456. CMD_SET_FB_ADDR_CB_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
CB_BASE_FBC_CBCR_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 1.457. 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

1.2.5.2.8.56. CMD_SET_FB_ADDR_CR_BASE7 (0x000001AC)

Cr base of index7

Table 1.458. 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 1.459. 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

1.2.5.2.8.57. CMD_SET_FB_ADDR_FBC_Y_OFFSET7 (0x000001AC)

FBC luma offset base of index7

Table 1.460. 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 1.461. 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

1.2.5.2.8.58. CMD_SET_FB_ADDR_FBC_C_OFFSET7 (0x000001B0)

FBC chroma offset base of index7

Table 1.462. 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 1.463. 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

1.2.5.2.8.59. CMD_SET_FB_ADDR_MV_COL0 (0x000001B4)

Colocated mv buffer base of index 0

Table 1.464. 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 1.465. CMD_SET_FB_ADDR_MV_COL0 Field Description

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

1.2.5.2.8.60. CMD_SET_FB_ADDR_MV_COL1 (0x000001B8)

Colocated mv buffer base of index 1

Table 1.466. CMD_SET_FB_ADDR_MV_COL1 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_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 1.467. CMD_SET_FB_ADDR_MV_COL1 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE1	R/W	Base address of colocated motion vecotors buffer of DPB index 1	0x0

1.2.5.2.8.61. CMD_SET_FB_ADDR_MV_COL2 (0x000001BC)

Colocated mv buffer base of index 2

Table 1.468. CMD_SET_FB_ADDR_MV_COL2 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_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 1.469. CMD_SET_FB_ADDR_MV_COL2 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE2	R/W	Base address of colocated motion vecotors buffer of DPB index 2	0x0

1.2.5.2.8.62. CMD_SET_FB_ADDR_MV_COL3 (0x000001C0)

Colocated mv buffer base of index 3

Table 1.470. 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 1.471. CMD_SET_FB_ADDR_MV_COL3 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE3	R/W	Base address of colocated motion vecotors buffer of DPB index 3	0x0

1.2.5.2.8.63. CMD_SET_FB_ADDR_MV_COL4 (0x000001C4)

Colocated mv buffer base of index 4

Table 1.472. 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

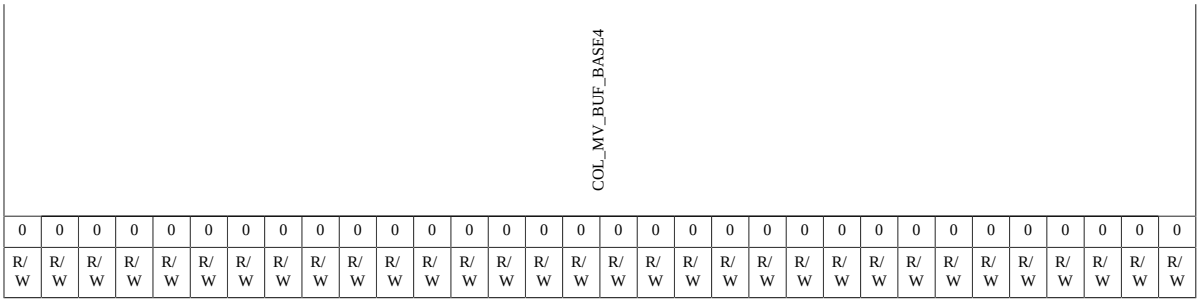


Table 1.473. CMD_SET_FB_ADDR_MV_COL4 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE4	R/W	Base address of colocated motion vecotors buffer of DPB index 4	0x0

1.2.5.2.8.64. CMD_SET_FB_ADDR_MV_COL5 (0x000001C8)

Colocated mv buffer base of index 5

Table 1.474. CMD_SET_FB_ADDR_MV_COL5 Bit Assignment

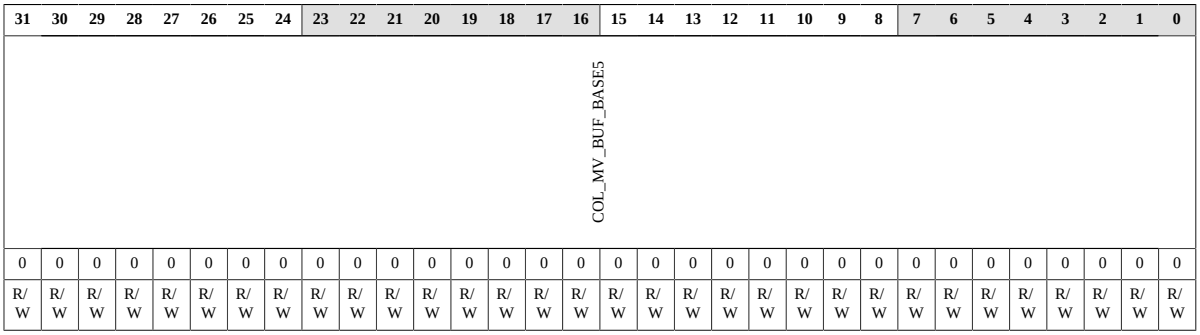


Table 1.475. CMD_SET_FB_ADDR_MV_COL5 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE5	R/W	Base address of colocated motion vecotors buffer of DPB index 5	0x0

1.2.5.2.8.65. CMD_SET_FB_ADDR_MV_COL6 (0x000001CC)

Colocated mv buffer base of index 6

Table 1.476. CMD_SET_FB_ADDR_MV_COL6 Bit Assignment

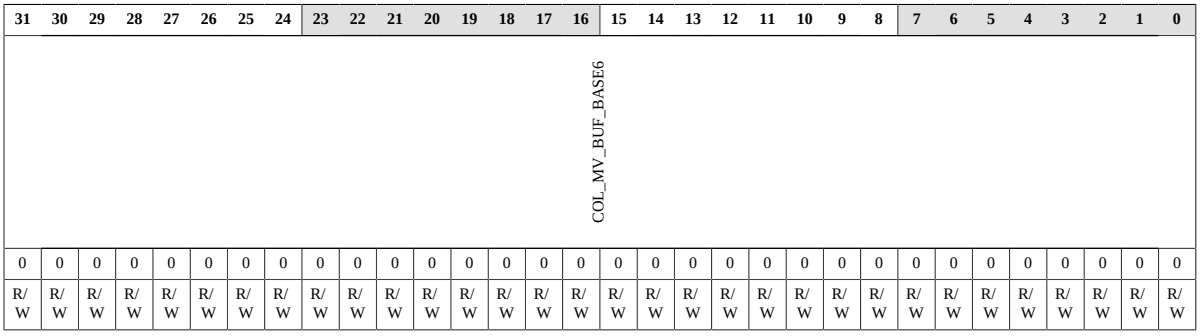


Table 1.477. CMD_SET_FB_ADDR_MV_COL6 Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COL_MV_BUF_BASE6	R/W	Base address of colocated motion vecotors buffer of DPB index 6	0x0

1.2.5.2.8.66. CMD_SET_FB_ADDR_MV_COL7 (0x000001D0)

Colocated mv buffer base of index 7

Table 1.478. CMD_SET_FB_ADDR_MV_COL7 Bit Assignment

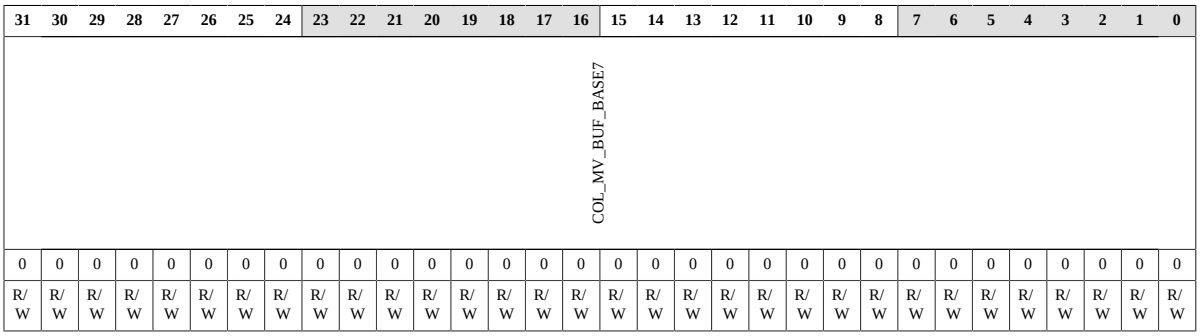


Table 1.479. CMD_SET_FB_ADDR_MV_COL7 Field Description

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

1.2.5.2.8.67. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.480. RET_BS_EMPTY Bit Assignment

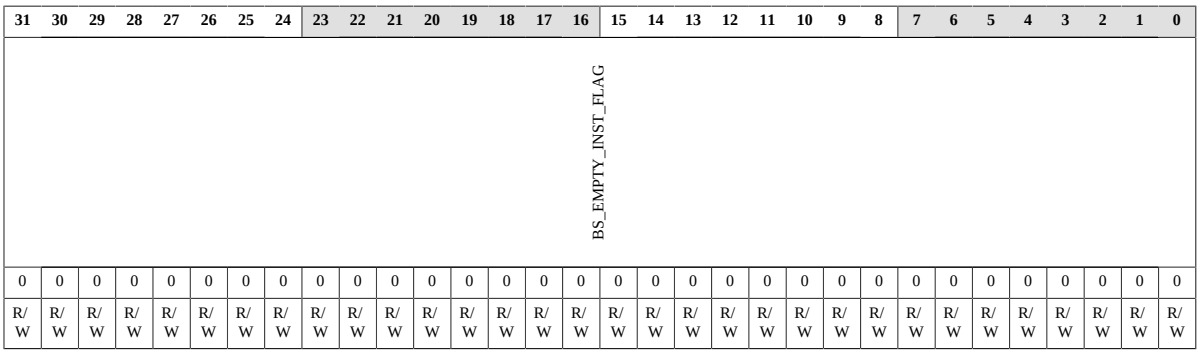


Table 1.481. RET_BS_EMPTY Field Description

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	0x0

Bit	Name	Type	Function	Reset Value
			can identify which instance's CPB is in empty status with instance_index. This field is ignored after BS_EMPTY interrupt service is done.	

1.2.5.2.8.68. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.482. 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 1.483. 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 its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.8.69. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.484. 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_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.485. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.8.70. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.486. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.8.71. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.488. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0

Bit	Name	Type	Function	Reset Value
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.8.72. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.490. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.9. DEC_PIC Command Parameter Registers

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

1.2.5.2.9.1. 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 **DEC_PIC** command (**0x0100** is given to this register).

Table 1.492. 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 1.493. 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 : **DEC_PIC** 0x4000 : QUERY 0x8000 : UPDATE_BS	0x0

1.2.5.2.9.2. CMD_DEC_PIC_OPTION (0x00000104)

DEC_PIC command option

Table 1.494. CMD_DEC_PIC_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
ENABLE_BW_OPT	RSVD																									HANDLE_CRA_AS_BLA		SKIP_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
RW	R	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	RW	RW	RW	RW	RW

Table 1.495. CMD_DEC_PIC_OPTION Field Description

Bit	Name	Type	Function	Reset Value
[31]	ENABLE_BW_OPT	RW	0: enable FBC output for non reference pic type when WTL is on.	0x0

Bit	Name	Type	Function	Reset Value
			1: disalbe FBC output for non reference pic when WTL is on. It is ignored when WTL is off.	
[30:6]	RSVD	R	Reserved	0x0
[5:4]	HANDLE_CRA_AS_BLA	RW	0x2: handle CRA picture as BLA. skip RASL pictures followd by CRA pictures	0x0
[3:0]	SKIP_MODE	RW	0x00: normal DEC_PIC 0x10: thumbnail mode (skip non-IRAP w/o ref.reg.) 0x11: skip non-IRAP 0x12: skip non-I picture 0x13: skip non-reference picture 0x14: skip picture whose temporal id > target temporal id(MAX_DEC_TEMP_ID).	0x0

1.2.5.2.9.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.496. 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 1.497. 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_WARNIG	0x0

1.2.5.2.9.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.498. 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 1.499. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.9.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.500. CMD_INSTANCE_INFO Bit Assignment

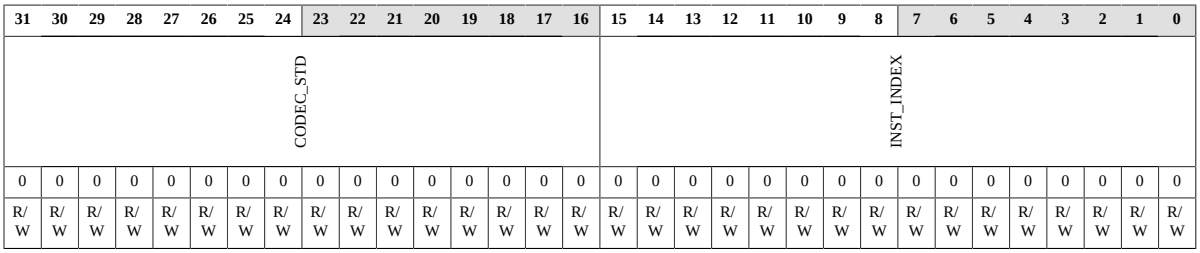


Table 1.501. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run HEVC_DEC = 0x0 VP9_DEC = 0x16	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.9.6. CMD_BS_RD_PTR (0x00000118)

Bistream Buffer Read Pointer

Table 1.502. CMD_BS_RD_PTR Bit Assignment

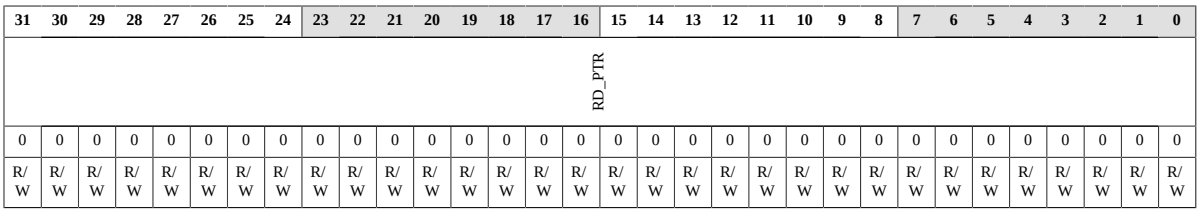


Table 1.503. CMD_BS_RD_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	RD_PTR	R/W	Start address of bitstream for handling current command. This is valid only if RD_PTR_VALID_FLAG is 1. If RD_PTR_VALID_FLAG is 0. VPU starts to decode from the AU end position of the last decoded picture in CPB. VPU also updates this (RET_ADDR_BS_RD_PTR) with end address of a NAL, when a NAL is decoded.	0x0

1.2.5.2.9.7. CMD_BS_WR_PTR (0x0000011C)

Bistream Buffer Write Pointer

Table 1.504. CMD_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
WR_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.505. CMD_BS_WR_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	WR_PTR	R/W	End address of bitstream for handling current command If a bitstream_empty interrupt is asserted, host processor should do either feed more bitstream and update WR_PTR or set EXPLICIT_END to complete decoding anyhow using UPDATE_BS QUERY command.	0x0

1.2.5.2.9.8. CMD_BS_OPTIONS (0x00000120)

Bistream buffer option

Table 1.506. CMD_BS_OPTIONS 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_VALID_FLAG	RSVD																													STREAM_END	EXPLICIT_END
0	0	0	0	0	0	0	0	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 1.507. CMD_BS_OPTIONS Field Description

Bit	Name	Type	Function	Reset Value
[31]	RD_PTR_VALID_FLAG	R/W	RD_PTR address valid flag If bitstream mode is PIC_END, this field should be 1.	0x0
[30:2]	RSVD	R/W	Reserved	0x0
[1]	STREAM_END	R/W	This field makes VPU assume Stream End when there is no stream to feed in the buffer.	0x0
[0]	EXPLICIT_END	R/W	Explicit End When this field is set to 1, VPU assumes that bitstream buffer has at least one single frame and follows the tasks below. 1. VPU decodes until the end of bitstream buffer (WR_PTR) even if the last 3 bytes are all 0. 2. VPU returns success if it has decoded a frame successfully.	0x0

Bit	Name	Type	Function	Reset Value
			If bitstream is insufficient to complete decoding a frame, VPU performs what it is supposed to do with the specified task in BS_SHORTAGE_OPTION of BS_PARAM. If this flag is 0, 1. VPU decodes to the almost end of bitstream buffer (WR_PTR), but not to some bytes (less than 3). It intentionally does not consume some a few bytes, because VPU is not sure if the last bytes are the start code of next NAL or they might not fill the offset in the CABAC. 2. VPU returns success if it has decoded a frame successfully. If bitstream is insufficient to complete decoding a frame, VPU stops decoding and waits for more bitstream to be filled. Then host processor can do either feed bitstream more or set EXPLICIT_END to complete decoding anyhow. BITSTREAM_EMPTY interrupt is asserted when EXPLITCT_END = 0 or when bitstream buffer is near empty (not real empty) for seamless decoding. **Caution** Host processor can set this register any time, but cannot clear during command processing.	

1.2.5.2.9.9. CMD_DEC_VCORE_LIMIT (0x00000124)

V-Core Limit

Table 1.508. CMD_DEC_VCORE_LIMIT 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																												VCORE_LIMIT			
0	0	0	0	0	0	0	0	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/W	R/W	R/W	R/W

Table 1.509. CMD_DEC_VCORE_LIMIT Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	VCORE_LIMIT	R/W	Limit the number of V-COREs to be used while decoding a picture. This valuse is only available when VPU is configured with multiple V-COREs.	0x0

1.2.5.2.9.10. CMD_SEQ_CHANGE_ENABLE_FLAG (0x00000128)

Sequence change enable flag

Table 1.510. CMD_SEQ_CHANGE_ENABLE_FLAG 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

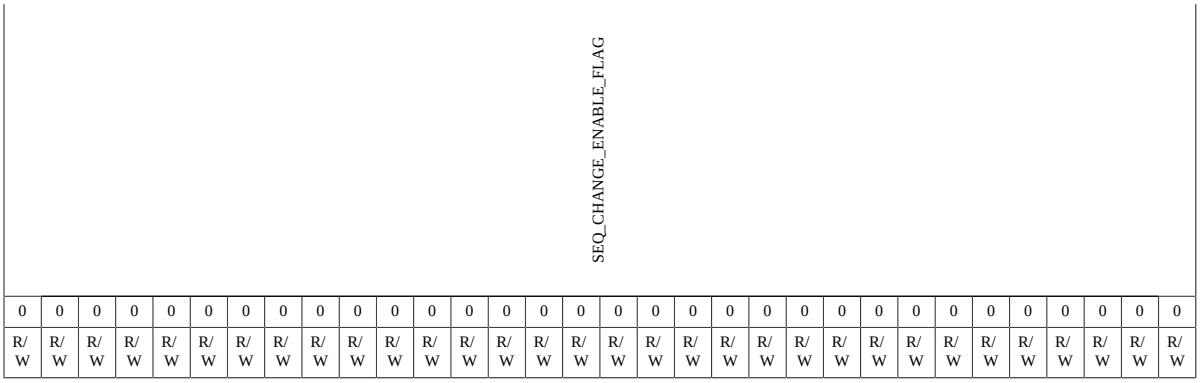


Table 1.511. CMD_SEQ_CHANGE_ENABLE_FLAG Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SEQ_CHANGE_ENABLE_FLAG	R/W	If any bit flag of SEQ_CHANGE_ENABLE_FLAG is 1 and the syntax value corresponding the bit flag is changed with a newly activated SPS, VPU regards it as sequence change. VPU reports it to SEQ_CHANGE_FLAG register, bumps out all of the pictures that remain in DPB, and clears the reference flag. SEQ_CHANGE_ENABLE_FLAG[4:0] : reserved SEQ_CHANGE_ENABLE_FLAG[5] : general_profile_idc SEQ_CHANGE_ENABLE_FLAG[15:6] : reserved SEQ_CHANGE_ENABLE_FLAG[16] : pic_width_in_luma_samples, pic_height_in_luma_samples SEQ_CHANGE_ENABLE_FLAG[18:17] : reserved SEQ_CHANGE_ENABLE_FLAG[19] : sps_max_dec_pic_buffering_minus[i], sps_max_num_reorder_pics[i], sps_max_latency_increase_plus1[i] SEQ_CHANGE_ENABLE_FLAG[31:20] : reserved For VP9 DEC, • SEQ_CHANGE_ENABLE_FLAG[15:0] : reserved • SEQ_CHANGE_ENABLE_FLAG[16] : pic_width_in_luma_samples, pic_height_in_luma_samples in key frame • SEQ_CHANGE_ENABLE_FLAG[17] : pic_width_in_luma_samples, pic_height_in_luma_samples in inter frame • SEQ_CHANGE_ENABLE_FLAG[31:18] : reserved	0x0

1.2.5.2.9.11. CMD_DEC_SEI_MASK (0x0000012C)

User data dump option

Table 1.512. CMD_DEC_SEI_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
SUFFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_2	SUFFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_1	PREFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_2	PREFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_1	RSVD											SUFFIX_SEI_COLOUR_REAMPPING_INFO	PREFIX_SEI_CONTENT_LIGHT_LEVEL_INFO	PREFIX_SEI_FILM_GRAIN_CHARACTERISTICS_INFO	PREFIX_SEI_TONE_MAPPING_INFO	PREFIX_SEI_KNEE_FUNCTION_INFO	PREFIX_SEI_CHROMA_RESAMPLING_FILTER_HINT	PREFIX_SEI_MASTERING_DISPLAY_COLOUR_VOLUME	RSVD	SUFFIX_SEI_USER_DATA_UNREGISTERED	SUFFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_0	PREFIX_SEI_USER_DATA_UNREGISTERED	PREFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_0	PREFIX_SEI_PIC_TIMING	RSVD	REPORT_VUI	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 1.513. CMD_DEC_SEI_MASK Field Description

Bit	Name	Type	Function	Reset Value
[31]	SUFFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_2	R/W	Enable to report the 3rd user_data_registered_itu_t_t35( ) SUFFIX_SEI message.	0x0
[30]	SUFFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_1	R/W	Enable to report the 2nd user_data_registered_itu_t_t35( ) SUFFIX_SEI message.	0x0
[29]	PREFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_2	R/W	Enable to report the 3rd user_data_registered_itu_t_t35( ) PREFIX_SEI message.	0x0
[28]	PREFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_1	R/W	Enable to report the 2nd user_data_registered_itu_t_t35( ) PREFIX_SEI message.	0x0
[27:17]	RSVD	R/W	Reserved	0x0
[16]	SUFFIX_SEI_COLOUR_REAMPPING_INFO	R/W	Enable to report colour_remapping_info( ) PREFIX_SEI message.	0x0

Bit	Name	Type	Function	Reset Value
[15]	PREFIX_SEI_CONTENT_LIGHT_LEVEL_INFO	R/W	Enable to report content_light_level_ info( ) PREFIX_SEI message.	0x0
[14]	PREFIX_SEI_FILM_GRAIN_CHARACTERISTICS_INFO	R/W	Enable to report film_grain_ char- acteristics_ info() message.	0x0
[13]	PREFIX_SEI_TONE_MAPPING_INFO	R/W	Enable to report tone_mapping_info PREFIX_SEI message.	0x0
[12]	PREFIX_SEI_KNEE_FUNCTION_INFO	R/W	Enable to report knee_function_ in- fo( ) PREFIX_SEI message.	0x0
[11]	PREFIX_SEI_CHROMA_RESAMPLING_FILTER_HINT	R/W	Enable to re- port chroma_ resampling_filter_ hint( ) PREFIX_SEI message.	0x0
[10]	PREFIX_SEI_MASTERING_DISPLAY_COLOUR_VOLUME	R/W	Enable to report mastering_display_ colour_volume( ) PREFIX_SEI message.	0x0
[9]	RSVD	R/W	Reserved	0x0
[8]	SUFFIX_SEI_USER_DATA_UNREGISTERED	R/W	Enable to re- port user_data_ unregistered( ) SUFFIX_SEI message.	0x0
[7]	SUFFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_0	R/W	Enable to report the 1st user_data_ registered_itu_t_ t35( ) SUFFIX_SEI message.	0x0
[6]	PREFIX_SEI_USER_DATA_UNREGISTERED	R/W	Enable to re- port user_data_ unregistered( ) PREFIX_SEI message.	0x0
[5]	PREFIX_SEI_USER_DATA_REGISTERED_ITU_T_T35_0	R/W	Enable to report the 1st user_data_ registered_itu_t_ t35( ) PREFIX_SEI message.	0x0
[4]	PREFIX_SEI_PIC_TIMING	R/W	Enable to report pic_timing( )	0x0

Bit	Name	Type	Function	Reset Value
			PREFIX_SEI message.	
[3]	RSVD	R/W	Reserved	0x0
[2]	REPORT_VUI	R/W	Enable to report VUI.	0x0
[1:0]	RSVD	R/W	Reserved	0x0

1.2.5.2.9.12. CMD_DEC_TEMPORAL_ID_PLUS1 (0x00000130)

Max Decode Temporal ID

Table 1.514. CMD_DEC_TEMPORAL_ID_PLUS1 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_DEC_TEMP_ID_PLUS1							
0	0	0	0	0	0	0	0	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 1.515. CMD_DEC_TEMPORAL_ID_PLUS1 Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	MAX_DEC_TEMP_ID_PLUS1	R/W	A max temporal ID to decode MAX_DEC_TEMP_ID = MAX_DEC_TEMP_ID_PLUS1 -1. If the value of MAX_DEC_TEMP_ID is equal to 0x0: it decodes a picture of all ranges of temporal ID. Temporal ID decoding constraint off. 0x1 ~ 0x6 : it decodes a picture if the temporal ID is less than or equal to MAX_DEC_TEMP_ID. Or discard a picture if its temporal ID is greater than MAX_DEC_TEMP_ID.	0x0

1.2.5.2.9.13. CMD_DEC_FORCE_FB_LATENCY_PLUS1 (0x00000134)

User define display latency

Table 1.516. CMD_DEC_FORCE_FB_LATENCY_PLUS1 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

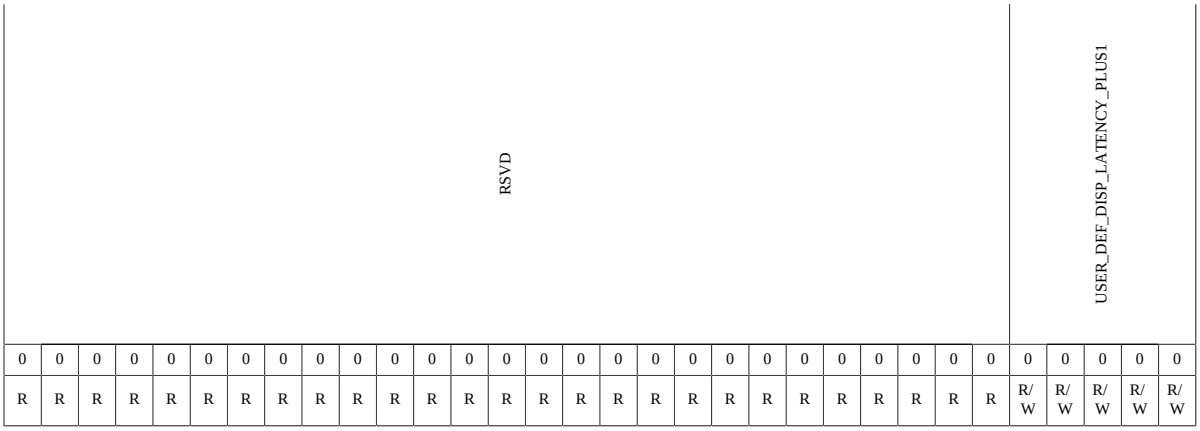


Table 1.517. CMD_DEC_FORCE_FB_LATENCY_PLUS1 Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	USER_DEF_DISP_LATENCY_PLUS1	R/W	Change frame buffer display latency by force. 0x0: not used 0x1~0x1f : latency + 1 (if this is set to 1, that means latency is 0 - immediate out)	0x0

1.2.5.2.9.14. CMD_DEC_USE_SEC_AXI (0x00000150)

Secondary AXI Usage option

Table 1.518. CMD_DEC_USE_SEC_AXI Bit Assignment

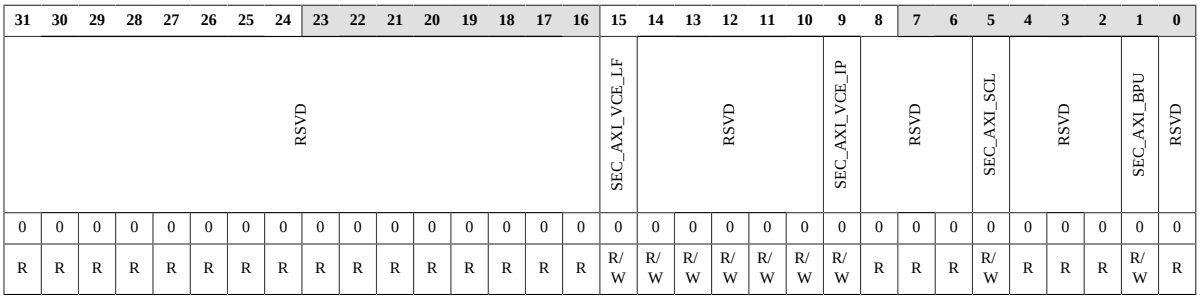


Table 1.519. CMD_DEC_USE_SEC_AXI Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15]	SEC_AXI_VCE_LF	R/W	Use 2nd AXI temp buffer for read channel of VCE LF row buffer.	0x0
[14:10]	RSVD	R/W	Reserved	0x0
[9]	SEC_AXI_VCE_IP	R/W	Use 2nd AXI temp buffer for read channel of VCE IP row buffer.	0x0
[8:6]	RSVD	R	Reserved	0x0
[5]	SEC_AXI_SCL	R/W	Use 2nd AXI temp buffer for read/write channel of Scaler line buffer. **Note** This field is valid only for use of Scaler.	0x0
[4:2]	RSVD	R	Reserved	0x0

Bit	Name	Type	Function	Reset Value
[1]	SEC_AXI_BPU	R/W	Use 2nd AXI temp buffer for Read channel of BPU NB row buffer.	0x0
[0]	RSVD	R	Reserved	0x0

1.2.5.2.9.15. RET_QUEUE_STATUS (0x000001E0)

Table 1.520. RET_QUEUE_STATUS Bit Assignment

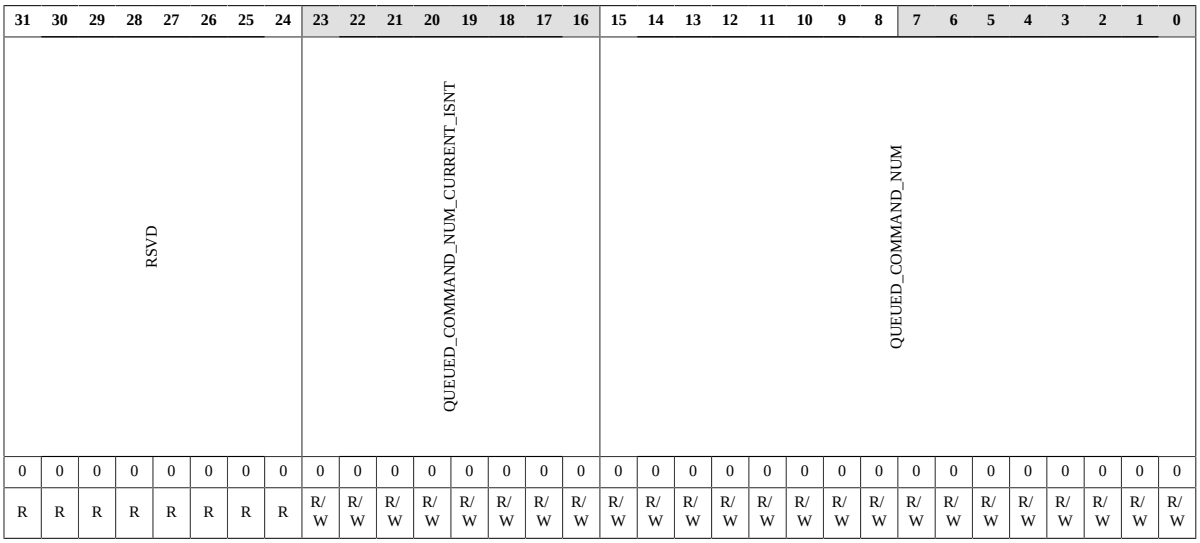


Table 1.521. RET_QUEUE_STATUS Field Description

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

1.2.5.2.9.16. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.522. RET_BS_EMPTY Bit Assignment

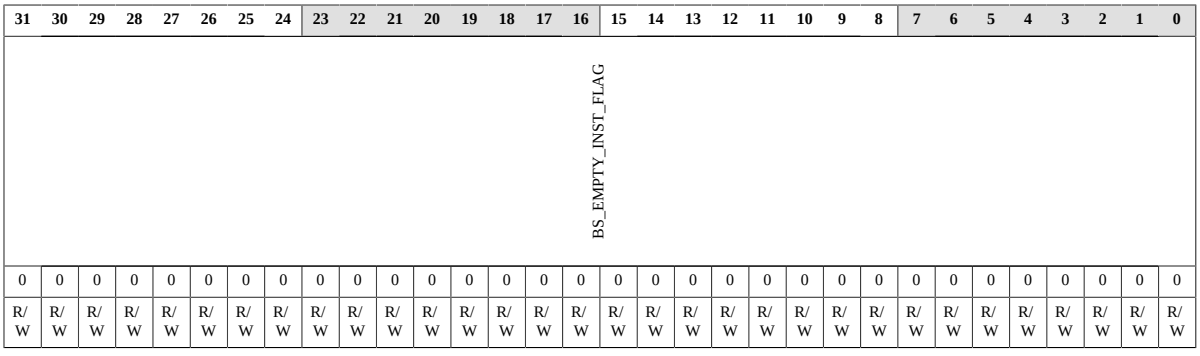


Table 1.523. RET_BS_EMPTY Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_EMPTY_INST_FLAG	R/W	BS_EMPTY_INST_FLAG = 1 << instance_index	0x0

Bit	Name	Type	Function	Reset Value
			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.	

1.2.5.2.9.17. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.524. RET_QUEUED_CMD_DONE Bit Assignment

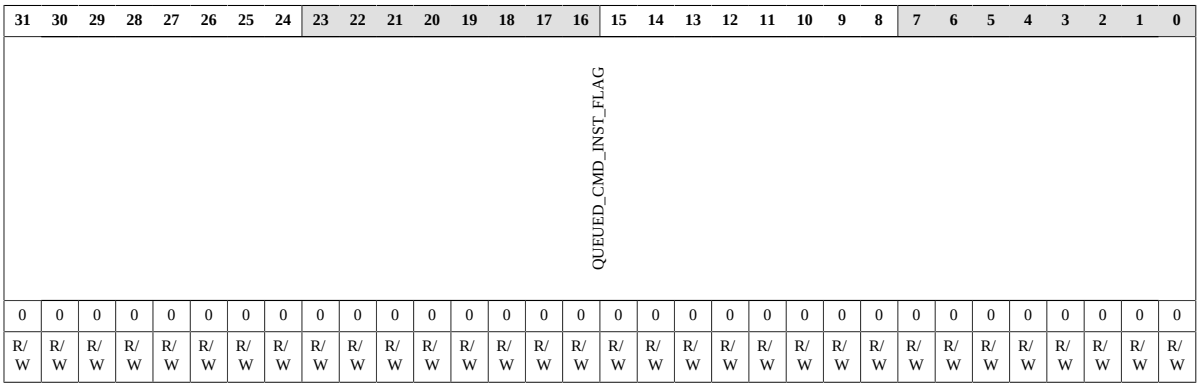


Table 1.525. RET_QUEUED_CMD_DONE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	QUEUED_CMD_INST_FLAG	R/W	QUEUED_CMD_INST_FLAG = 1 << instance_index When VPU completes its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.9.18. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.526. RET_SEEK_INSTANCE_INFO Bit Assignment

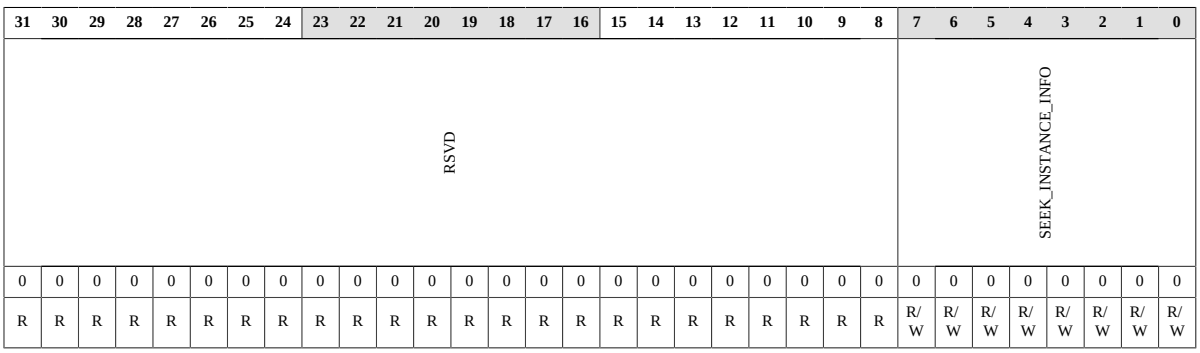


Table 1.527. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.9.19. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.528. RET_PARSING_INSTANCE_INFO Bit Assignment

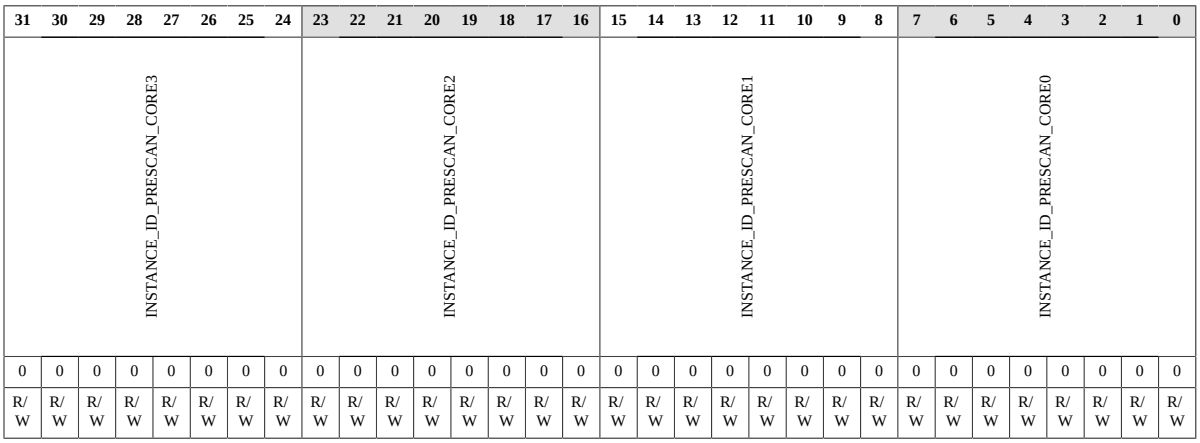


Table 1.529. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.9.20. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.530. RET_DECODING_INSTANCE_INFO Bit Assignment

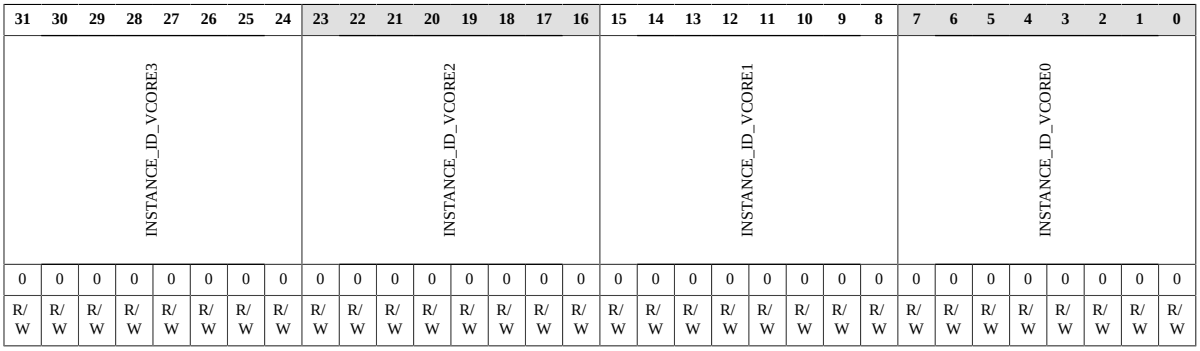


Table 1.531. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.9.21. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.532. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.533. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.10. QUERY(GET_VPU_INFO) Command Parameter Registers

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

1.2.5.2.10.1. 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 **QUERY** command (**0x4000** is given to this register).

Table 1.534. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : **QUERY** 0x8000 : UPDATE_BS	0x0

1.2.5.2.10.2. CMD_QUERY_OPTION (0x00000104)

Table 1.536. 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 1.537. 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 0x01: SET_WRITE_PROTECTION	0x0

Bit	Name	Type	Function	Reset Value
			0x02: GET_RESULT 0x03: UPDATE_DISP_IDC	

1.2.5.2.10.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.538. 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 1.539. 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_WARNIG	0x0

1.2.5.2.10.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.540. 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 1.541. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.10.5. RET_QUERY_FW_VERSION (0x00000118)

Table 1.542. 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

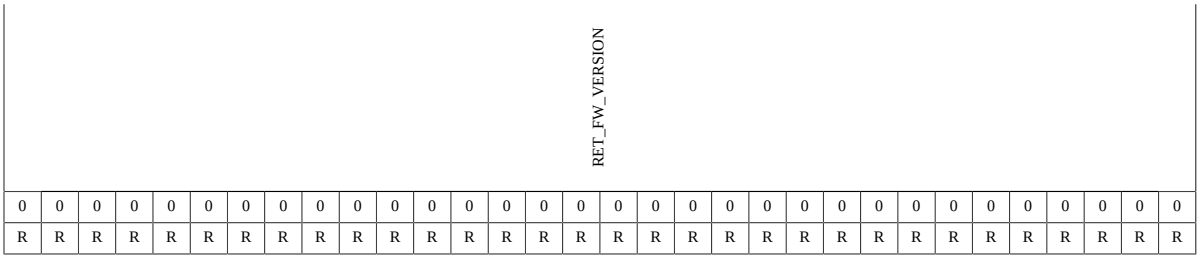


Table 1.543. RET_QUERY_FW_VERSION Field Description

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

1.2.5.2.10.6. RET_QUERY_PRODUCT_NAME (0x0000011C)

Table 1.544. RET_QUERY_PRODUCT_NAME Bit Assignment

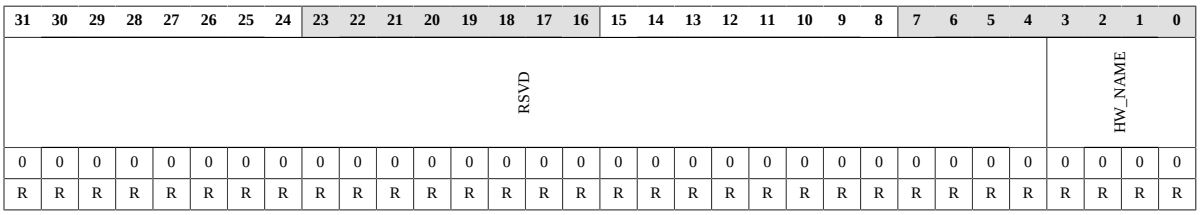


Table 1.545. RET_QUERY_PRODUCT_NAME Field Description

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".	0x0

1.2.5.2.10.7. RET_QUERY_PRODUCT_VERSION (0x00000120)

Table 1.546. RET_QUERY_PRODUCT_VERSION Bit Assignment

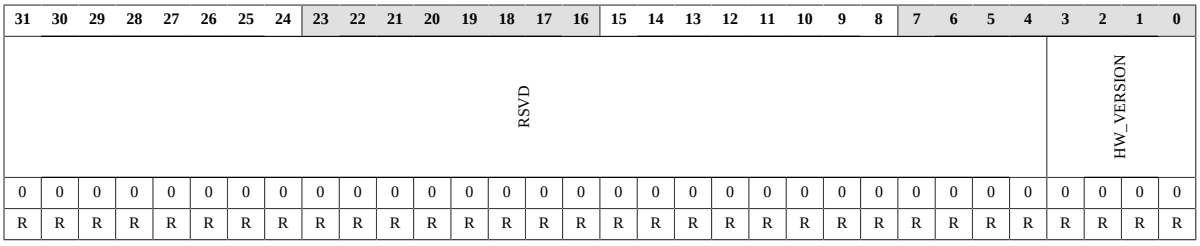


Table 1.547. RET_QUERY_PRODUCT_VERSION Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	HW_VERSION	R	VPU hardware product version It always returns "5120" for WAVE512 product..	0x0

1.2.5.2.10.8. RET_QUERY_STD_DEF0 (0x00000124)

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

Table 1.548. RET_QUERY_STD_DEF0 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
MAP_CONVERTER_REG	MAP_CONVERTER_SIG	CONFIG_INFO														AFBC_EN				FBC_EN				SCALER_EN				BWB_EN	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 1.549. 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

1.2.5.2.10.9. RET_QUERY_STD_DEF1 (0x00000128)

General Hardware Configuration Information (internal use only)

Table 1.550. 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 1.551. RET_QUERY_STD_DEF1 Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	INTERNAL_BLOCK	R	Internal block	0x0

Bit	Name	Type	Function	Reset Value
			[31:24] Enabled Support Logics (CU_REPORT, GCU etc.) [26]: MULTICORE_EN [25]: GCU_EN [24]: CU_REPORT [23:16] MULTICORE_PRESENT_INFO	
[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

1.2.5.2.10.10. RET_QUERY_CONF_FEATURE (0x0000012C)

Table 1.552. 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
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 1.553. RET_QUERY_CONF_FEATURE Field Description

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

1.2.5.2.10.11. RET_QUERY_CONF_DATE (0x00000130)

Table 1.554. 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 1.555. 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

1.2.5.2.10.12. RET_QUERY_CONF_REVISION (0x00000134)

Table 1.556. 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

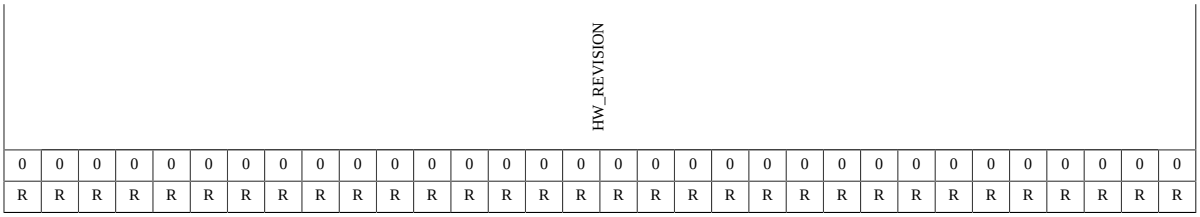


Table 1.557. RET_QUERY_CONF_REVISION Field Description

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

1.2.5.2.10.13. RET_QUERY_CONF_TYPE (0x00000138)

Table 1.558. RET_QUERY_CONF_TYPE Bit Assignment

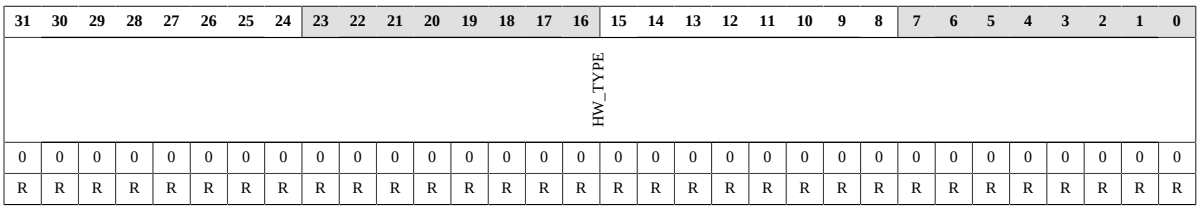


Table 1.559. RET_QUERY_CONF_TYPE Field Description

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

1.2.5.2.10.14. RET_QUERY_PRODUCT_ID (0x0000013C)

Table 1.560. RET_QUERY_PRODUCT_ID Bit Assignment

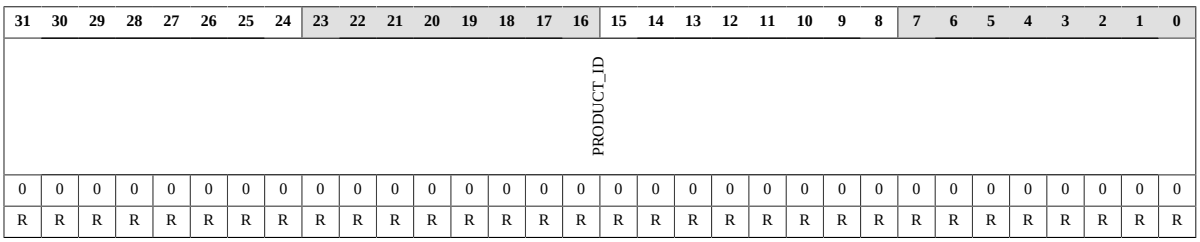
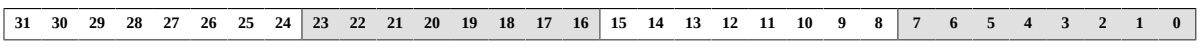


Table 1.561. RET_QUERY_PRODUCT_ID Field Description

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

1.2.5.2.10.15. RET_QUERY_CUSTOMER_ID (0x00000140)

Table 1.562. RET_QUERY_CUSTOMER_ID Bit Assignment



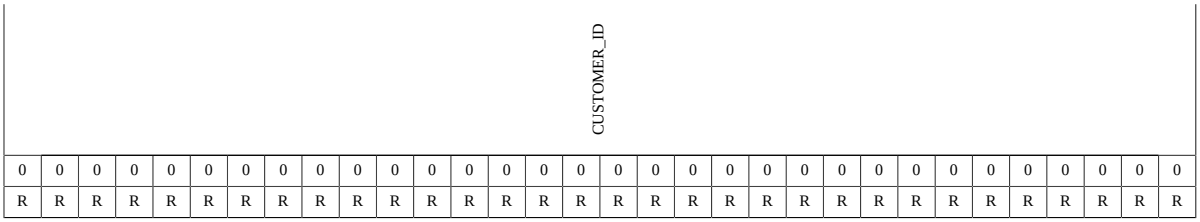


Table 1.563. RET_QUERY_CUSTOMER_ID Field Description

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

1.2.5.2.10.16. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.564. RET_BS_EMPTY Bit Assignment

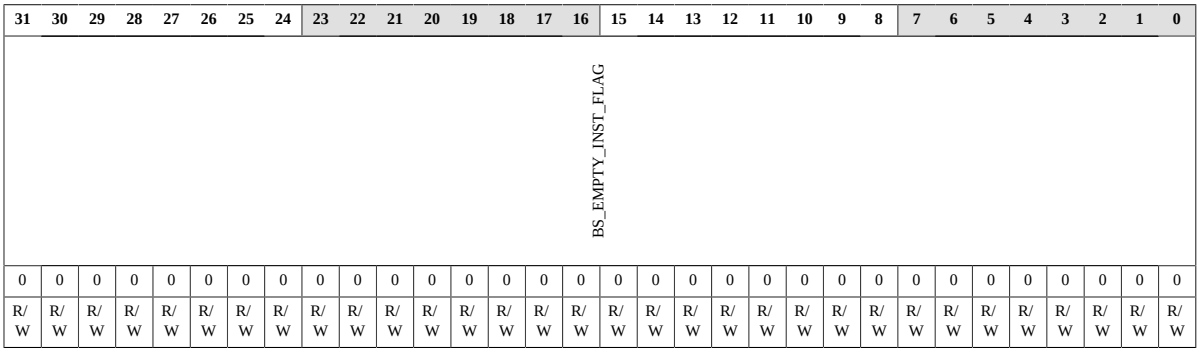
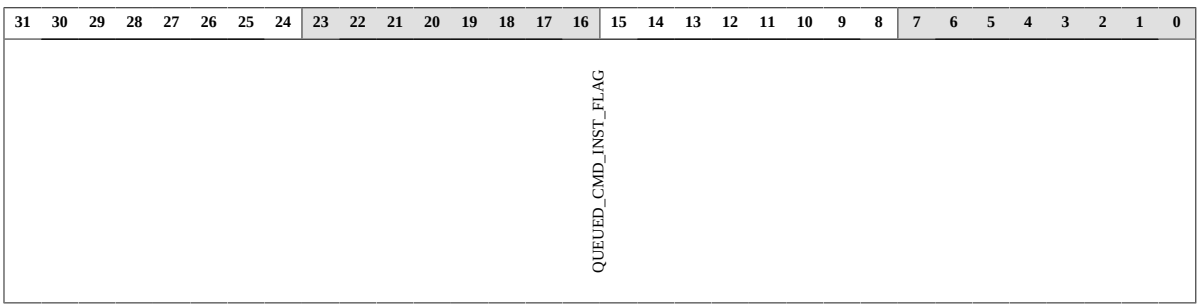


Table 1.565. RET_BS_EMPTY Field Description

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.	0x0

1.2.5.2.10.17. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.566. RET_QUEUED_CMD_DONE 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 1.567. 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 its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.10.18. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.568. 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_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.569. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.10.19. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.570. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.571. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.10.20. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.572. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.573. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.10.21. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.574. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							

0	0	0	0	0	0	0	0	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 1.575. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.11. QUERY(GET_RESULT) Command Parameter Registers

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

1.2.5.2.11.1. 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 **QUERY** command (**0x4000** is given to this register).

Table 1.576. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.577. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : **QUERY** 0x8000 : UPDATE_BS	0x0

1.2.5.2.11.2. CMD_QUERY_OPTION (0x00000104)

Table 1.578. 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 1.579. 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 0x01: SET_WRITE_PROTECTION 0x02: GET_RESULT 0x03: UPDATE_DISP_IDC	0x0

1.2.5.2.11.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.580. 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 1.581. 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_WARNIG	0x0

1.2.5.2.11.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.582. 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 1.583. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.11.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.584. 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 1.585. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run HEVC_DEC = 0x0 VP9_DEC = 0x16	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.11.6. CMD_DEC_ADDR_USER_BASE (0x00000114)

User Data Buffer Base Address

Table 1.586. CMD_DEC_ADDR_USER_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
USER_DATA_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 1.587. CMD_DEC_ADDR_USER_BASE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	USER_DATA_BUF_BASE	R/W	Base address of user data buffer User data buffer holds SEI, SPS, and VUI information. This address should be aligned in 4K boundary. VPU reports user data of the decoded frame if user data exists in the decoded frame and the relevant user data report option is enabled. User data buffering/reordering is required to sync user data with display frame.	0x0

1.2.5.2.11.7. CMD_DEC_USER_SIZE (0x00000118)

User Data Buffer Size

Table 1.588. CMD_DEC_USER_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
USER_DATA_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 1.589. CMD_DEC_USER_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	USER_DATA_BUF_SIZE	R/W	Size of user data buffer	0x0

1.2.5.2.11.8. CMD_DEC_USER_PARAM (0x0000011C)

User Data Buffer Parameter

Table 1.590. CMD_DEC_USER_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																												USER_DATA_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	R	R	R	R/W	R/W	R/W	R/W

Table 1.591. CMD_DEC_USER_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:4]	RSVD	R	Reserved	0x0
[3:0]	USER_DATA_ENDIAN	R/W	Endianness of user data	0x0

1.2.5.2.11.9. RET_QUERY_DEC_BS_RD_PTR (0x0000011C)

Bitstream buffer read pointer

Table 1.592. RET_QUERY_DEC_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
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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.593. RET_QUERY_DEC_BS_RD_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	BS_RD_PTR	R/W	VPU read address in the CPB. Host can feed bitstream data up to this address.	0x0

1.2.5.2.11.10. RET_QUERY_DEC_SEQ_PARAM (0x00000120)

Profile/Level/Tier/Max Sub layer

Table 1.594. RET_QUERY_DEC_SEQ_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
PROFILE_SPACE		PROFILE_TIER_FLAG	PROFILE_IDC					SPS_MAX_SUB_LAYER			RSVD	PROFILE_COMPATIBILITY_FLAG								PROGRESS_SOURCE_FLAG	INTERLACE_SOURCE_FLAG	NON_PACKED_CONSTRAINT_FLAG	FRAME_ONLY_CONSTRAINT_FLAG	LEVEL_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	RW	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.595. RET_QUERY_DEC_SEQ_PARAM Field Description

Bit	Name	Type	Function	Reset Value
[31:30]	PROFILE_SPACE	R/W	The value of general_profile_space of active SPS of H.265 See H.265 annex A for detail information.	0x0
[29]	PROFILE_TIER_FLAG	R/W	The value of general_tire_flag of active SPS of H.265	0x0
[28:24]	PROFILE_IDC	R/W	The value of general_profile_idc of active SPS of H.265 or a profile of VP9	0x0
[23:21]	SPS_MAX_SUB_LAYER	R/W	The value of maximum number of temporal sub-layers, sps_max_sub_layer_minus1+1, of active SPS of H.265	0x0
[20]	RSVD	RW	Reserved	0x0
[19:12]	PROFILE_COMPATIBILITY_FLAG	R/W	The value of first 8 bits out of general_profile_compatiblity_flag[32] of active SPS of H.265 PROFILE_COMPATIBILITY_FLAG[i] is equal to general_profile_compatiblity_flag[i] with i = 0 to 7.	0x0
[11]	PROGRESS_SOURCE_FLAG	R/W	The value of general_progressive_source_flag of active SPS of H.265	0x0
[10]	INTERLACE_SOURCE_FLAG	R/W	The value of general_interlaced_source_flag of active SPS of H.265	0x0
[9]	NON_PACKED_CONSTRAINT_FLAG	R/W	The value of general_non_packed_constraint_flag of active SPS of H.265	0x0
[8]	FRAME_ONLY_CONSTRAINT_FLAG	R/W	The value of general_frame_only_constraint_flag of active SPS of H.265	0x0
[7:0]	LEVEL_IDC	R/W	The value of general_level_idc of active SPS of H.265	0x0

1.2.5.2.11.11. RET_DEC_COLOR_SAMPLE_INFO (0x00000124)

Color Sample Information

Table 1.596. RET_DEC_COLOR_SAMPLE_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								ASPECT_RATIO_IDC								RSVD			LF_PIC_DBK_DISABLE	COLOR_FORMAT_IDC				BIT_DEPTH_CHROMA				BIT_DEPTH_LUMA			
0	0	0	0	0	0	0	0	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	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

Table 1.597. RET_DEC_COLOR_SAMPLE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	RSVD	R	Reserved	0x0
[23:16]	ASPECT_RATIO_IDC	R/W	The value of aspect_ratio_idc of H.265. See Table E-1 of Annex E.2 of VUI of H.265. If not decoded in the bitstream, the value of ASPECT_RATIO_IDC is equal to 0.	0x0
[15:13]	RSVD	R	Reserved	0x0
[12]	LF_PIC_DBK_DISABLE	R/W	H.265: 1 when both sao and loop filter are off. 0 otherwise. VP9: 1 when loop filter are off. 0 otherwise.	0x0
[11:8]	COLOR_FORMAT_IDC	R/W	Chroma sampling, chroma_format_idc. See H.265 spec clause 6.2 for more information	0x0
[7:4]	BIT_DEPTH_CHROMA	R/W	Bit depth of chroma sample	0x0
[3:0]	BIT_DEPTH_LUMA	R/W	Bit depth of luma sample	0x0

1.2.5.2.11.12. RET_DEC_ASPECT_RATIO (0x00000128)

Sample Aspect Ratio

Table 1.598. RET_DEC_ASPECT_RATIO 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
SAR_WIDTH																SAR_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 1.599. RET_DEC_ASPECT_RATIO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	SAR_WIDTH	R/W	The horizontal size of the sample aspect ratio. If not decoded in bitstream (VUI in case of H.265), the value of SarWidth is equal to 0.	0x0
[15:0]	SAR_HEIGHT	R/W	The vertical size of the sample aspect ratio.	0x0

Bit	Name	Type	Function	Reset Value
			If not decoded in bitstream(VUI in case of H.265), the value of SarHeight is equal to 0.	

1.2.5.2.11.13. RET_DEC_BIT_RATE (0x0000012C)

Maximum Bit Rate

Table 1.600. RET_DEC_BIT_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
MAX_BIT_RATE																															
0	0	0	0	0	0	0	0	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 1.601. RET_DEC_BIT_RATE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	MAX_BIT_RATE	R/W	The maximum input bit rate for maxNumSubLayer, as specified in sub layer HRD parameter of H.265.	0x0

1.2.5.2.11.14. RET_DEC_FRAME_RATE_NR (0x00000130)

Frame Rate Numerator

Table 1.602. RET_DEC_FRAME_RATE_NR 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
FRAME_RATE_NR																															
0	0	0	0	0	0	0	0	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 1.603. RET_DEC_FRAME_RATE_NR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FRAME_RATE_NR	R/W	This field is valid for H.265. Frame rate numerator If frame rate syntax is not decoded in bitstream, the value of FrameRateNr is equal to -1. If FRAME_RATE_NR and FRAME_RATE_DR are not-zero values, FrameRate is derived as follows. FrameRate =FRAME_RATE_DR/FRAME_RATE_NR. Otherwise if FRAME_RATE_NR or FRAME_RATE_DR are zero values, the value of FrameRate is invalid.	0x0

1.2.5.2.11.15. RET_DEC_FRAME_RATE_DR (0x00000134)

Frame Rate Denominator

Table 1.604. RET_DEC_FRAME_RATE_DR 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
FRAME_RATE_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 1.605. RET_DEC_FRAME_RATE_DR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FRAME_RATE_DR	R/W	This field is valid for H.265. Frame rate denominator If frame rate syntax is not decoded in bitstream, the value of FrameRateDr is equal to -1.	0x0

1.2.5.2.11.16. RET_QUERY_DEC_NUM_REQURED_FB (0x00000138)

Required Number of Minimum DPB

Table 1.606. RET_QUERY_DEC_NUM_REQURED_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																											MIN_DPB_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	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 1.607. RET_QUERY_DEC_NUM_REQURED_FB Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	MIN_DPB_NUM	R/W	Required number of frame buffers for decoding sequence	0x0

1.2.5.2.11.17. RET_QUERY_DEC_NUM_REORDER_DELAY (0x0000013C)

Reorder frame num.

Table 1.608. RET_QUERY_DEC_NUM_REORDER_DELAY 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																											REORDER_DELAY_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	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 1.609. RET_QUERY_DEC_NUM_REORDER_DELAY Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	REORDER_DELAY_NUM	R/W	The value of REORDER_DELAY_NUM is reorder picture number of H.265. No picture reordering in VP9.	0x0

1.2.5.2.11.18. RET_QUERY_DEC_SUB_LAYER_INFO (0x00000140)

Sub-layer

Table 1.610. RET_QUERY_DEC_SUB_LAYER_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																MAX_SUB_LAYER								TEMPORAL_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/W	R/W	R/W	R/W	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 1.611. RET_QUERY_DEC_SUB_LAYER_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	RSVD	R	Reserved	0x0
[15:8]	MAX_SUB_LAYER	R/W	Maximum number of sub-layers of H.265	0x0
[7:0]	TEMPORAL_ID	R/W	TemporalId of H.265	0x0

1.2.5.2.11.19. RET_QUERY_DEC_NOTIFICATION (0x00000144)

Sequence change flag

Table 1.612. RET_QUERY_DEC_NOTIFICATION 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														INTER_FRAME_RESOLUTION_CHANGE		RSVD												TIER_CHANGE	PROFILE_CHANGE	DPB_SIZE_CHANGE	SEQ_CHANGE_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	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W

Table 1.613. RET_QUERY_DEC_NOTIFICATION Field Description

Bit	Name	Type	Function	Reset Value
[31:18]	RSVD	R	Reserved	0x0
[17]	INTER_FRAME_RESOLUTION_CHANGE	R/W	If INTER_FRAME_RESOLUTION _CHANGE is equal to 1, a new DPB needs to be allocated for next picture decoding. (VP9 decoder only)	0x0
[16:4]	RSVD	R	Reserved	0x0
[3]	TIER_CHANGE	R/W	If TIER_CHANGE is equal to 1, the tier of the current picture is different from the tier of the previously decoded picture. Otherwise, the tier of the current picture is equal to the tier of the previously decoded picture.	0x0
[2]	PROFILE_CHANGE	R/W	If PROFILE_CHANGE is equal to 1, the profile of the current picture is different from the profile of the previously decoded picture. Otherwise, the profile of the current picture is equal to the profile of the previously decoded picture.	0x0
[1]	DPB_SIZE_CHANGE	R/W	If DPB_SIZE_CHANGE is equal to 1, the decoded size of the current picture is different from the size of the previously decoded picture. Otherwise, the decoded size of current picture is equal to the size of previously decoded picture.	0x0
[0]	SEQ_CHANGE_FLAG	R/W	If SEQ_CHANGE_FLAG is equal to 1, VPU detects a new sequence. It may require different decoding resources such as DPB, DPB number, and etc. Otherwise, VPU decodes the subsequent AU of the sequence.	0x0

1.2.5.2.11.20. RET_QUERY_DEC_USERDATA_IDC (0x00000148)

Table 1.614. RET_QUERY_DEC_USERDATA_IDC 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
USERDATA_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 1.615. RET_QUERY_DEC_USERDATA_IDC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	USERDATA_IDC	R/W	Each bit field of USER_DATA_FLAG specifies the report result to USER_DATA_BUF_BASE. USER_DATA_FLAG[0] is reserved	0x0

Bit	Name	Type	Function	Reset Value
			USER_DATA_FLAG[1] is flag of user data buffer full. USER_DATA_FLAG[2] is flag of vui paramter. USER_DATA_FLAG[3] is reserved USER_DATA_FLAG[4] is flag of pic_timing sei. USER_DATA_FLAG[5] is flag of the 1st user_data_registered_itu_t_t35 prefix sei. USER_DATA_FLAG[6] is flag of user_data_unregistered prefix sei. USER_DATA_FLAG[7] is flag of the 1st user_data_registered_itu_t_t35 suffix sei. USER_DATA_FLAG[8] is flag of user_data_unregistered suffix sei. USER_DATA_FLAG[9] is reserved. USER_DATA_FLAG[10] is flag of mastering_display_color_volume prefix sei. USER_DATA_FLAG[11] is flag of chroma_resampling_filter_hint prefix sei. USER_DATA_FLAG[12] is flag of knee_function_info sei. USER_DATA_FLAG[13] is flag of tone_mapping_info prefix sei. USER_DATA_FLAG[14] is flag of film_grain_characteristics_info prefix sei. USER_DATA_FLAG[15] is flag of content_light_level_info prefix sei. USER_DATA_FLAG[16] is flag of colour_remapping_info prefix sei. USER_DATA_FLAG[27:17] is reserved USER_DATA_FLAG[28] is flag of the 2nd user_data_registered_itu_t_t35 prefix sei. USER_DATA_FLAG[29] is flag of the 3rd user_data_registered_itu_t_t35 prefix sei. USER_DATA_FLAG[30] is flag of the 2nd user_data_registered_itu_t_t35 suffix sei. USER_DATA_FLAG[31] is flag of the 3rd user_data_registered_itu_t_t35 suffix sei. User data buffer full flag equal to 1 specifies that decoded frame has more user data size than VPU internal buffer. VPU only dumps the internal buffer size of user data to USER_DATA_BUF_BASE buffer. In other words, VPU is unable to report the rest of the user data to USER_DATA_BUF_BASE buffer after the internal buffer fullness happens.	

1.2.5.2.11.21. RET_QUERY_DEC_PIC_SIZE (0x0000014C)

Decoded Picture Size

Table 1.616. RET_QUERY_DEC_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
DECODED_PIC_WIDTH																DECODED_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 1.617. RET_QUERY_DEC_PIC_SIZE Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	DECODED_PIC_WIDTH	R/W	Decoded Picture Width The value of DECODED_PIC_WIDTH is eqaul to the value of pic_width_in_luma_samples in H.265 active SPS.	0x0
[15:0]	DECODED_PIC_HEIGHT	R/W	Decoded Picture Height The value of DECODED_PIC_HEIGHT is eqaul to the value of pic_height_in_luma_samples in H.265 active SPS.	0x0

1.2.5.2.11.22. RET_QUERY_DEC_CROP_TOP_BOTTOM (0x00000150)

Display Crop Offset Top/Bottom

Table 1.618. RET_QUERY_DEC_CROP_TOP_BOTTOM 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
DISPLAY_TOP_OFFSET																DISPLAY_BOTTOM_OFFSET															
0	0	0	0	0	0	0	0	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 1.619. RET_QUERY_DEC_CROP_TOP_BOTTOM Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	DISPLAY_TOP_OFFSET	R/W	Display top offset DISPLAY_TOP_OFFSET is equal to leftOffset. topOffset = conf_win_top_offset + def_disp_win_top_offset. See Annex E.3 of H.265 spec for more.	0x0
[15:0]	DISPLAY_BOTTOM_OFFSET	R/W	Display bottom offset DISPLAY_BOTTOM_OFFSET is equal to botto-mOffset. bottomOffset = conf_win_bottom_offset + def_disp_win_bottom_offset	0x0

1.2.5.2.11.23. RET_QUERY_DEC_CROP_LEFT_RIGHT (0x00000154)

Display Crop Offset Left/Right

Table 1.620. RET_QUERY_DEC_CROP_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

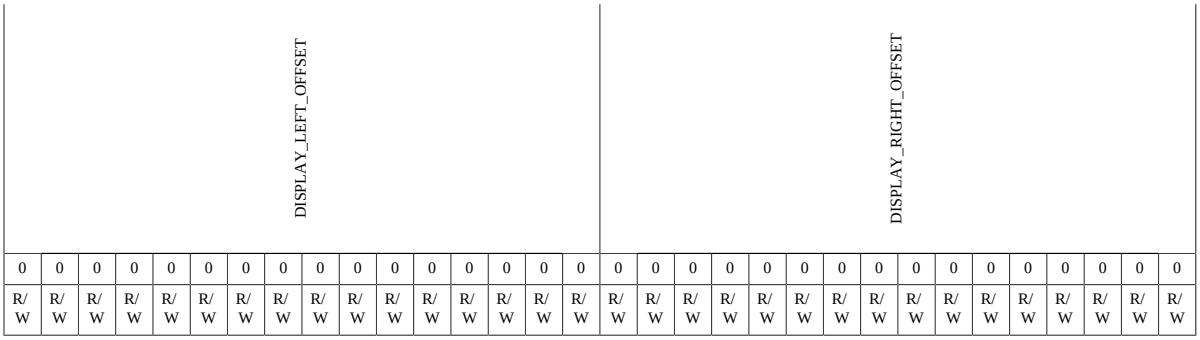


Table 1.621. RET_QUERY_DEC_CROP_LEFT_RIGHT Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	DISPLAY_LEFT_OFFSET	R/W	Display left offset DISPLAY_LEFT_OFFSET is equal to leftOffset. leftOffset = conf_win_left_offset + def_disp_win_left_offset	0x0
[15:0]	DISPLAY_RIGHT_OFFSET	R/W	Display right offset DISPLAY_RIGHT_OFFSET is equal to rightOffset. rightOffset = conf_win_right_offset + def_disp_win_right_offset	0x0

1.2.5.2.11.24. RET_QUERY_DEC_AU_START_POS (0x00000158)

AU Bitstream Start Position

Table 1.622. RET_QUERY_DEC_AU_START_POS Bit Assignment

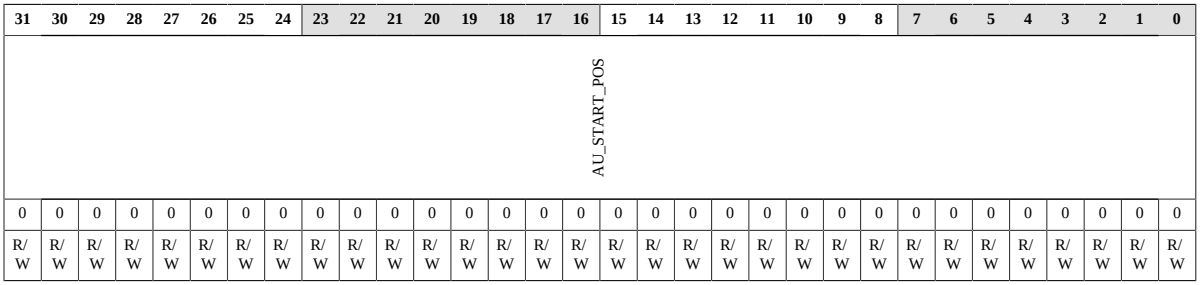


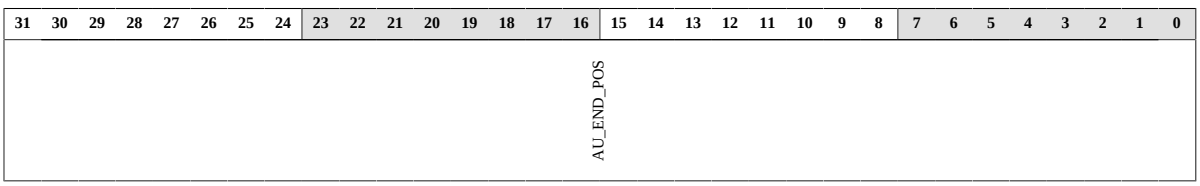
Table 1.623. RET_QUERY_DEC_AU_START_POS Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	AU_START_POS	R/W	Access unit(AU) bitstream start position	0x0

1.2.5.2.11.25. RET_QUERY_DEC_AU_END_POS (0x0000015C)

AU Bitstream End Position

Table 1.624. RET_QUERY_DEC_AU_END_POS 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 1.625. RET_QUERY_DEC_AU_END_POS Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	AU_END_POS	R/W	Access unit(AU) bitstream end position	0x0

1.2.5.2.11.26. RET_QUERY_DEC_PIC_TYPE (0x00000160)

Decoded picture type

Table 1.626. RET_QUERY_DEC_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
OUTPUT_FLAG	RSVD																			CTU_SIZE		VCL_NAL_UNIT_TYPE						RSVD	B_SLICE_DECODED	P_SLICE_DECODED	I_SLICE_DECODED
0	0	0	0	0	0	0	0	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	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	RW	R/W	R/W	R/W

Table 1.627. RET_QUERY_DEC_PIC_TYPE Field Description

Bit	Name	Type	Function	Reset Value
[31]	OUTPUT_FLAG	R/W	It is a picture output flag of H.265 while it is show_frame of VP9. If OUTPUT_FLAG is 0, the current decoded picture is not output for display. If OUTPUT_FLAG is 1, the current decoded picture is (will be) output for display.	0x0
[30:12]	RSVD	R	Reserved	0x0
[11:10]	CTU_SIZE	R/W	CTU size 0 : 16x16 1 : 32x32 2 : 64x64	0x0
[9:4]	VCL_NAL_UNIT_TYPE	R/W	VCL NAL unit type	0x0
[3]	RSVD	RW	Reserved	0x0
[2]	B_SLICE_DECODED	R/W	If more than one B slice is decoded, the value of B_SLICE_DECODED is equal to 1. Otherwise, if none of B slice is decoded, the value of B_SLICE_DECODED is equal to 0.	0x0
[1]	P_SLICE_DECODED	R/W	If more than one P slice is decoded, the value of P_SLICE_DECODED is equal to 1. Otherwise, if none of P slice is decoded, the value of P_SLICE_DECODED is equal to 0.	0x0
[0]	I_SLICE_DECODED	R/W	If more than one I slice is decoded, the value of I_SLICE_DECODED is equal to 1. Otherwise, if none of I slice is decoded, the value of I_SLICE_DECODED is equal to 0.	0x0

1.2.5.2.11.27. RET_QUERY_DEC_PIC_POC (0x00000164)

Picture Order Count

Table 1.628. RET_QUERY_DEC_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_ORDER_COUNT																															
0	0	0	0	0	0	0	0	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 1.629. RET_QUERY_DEC_PIC_POC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PIC_ORDER_COUNT	R/W	Picture order count of H.265	0x0

1.2.5.2.11.28. RET_QUERY_DEC_RECOVERY_POINT (0x00000168)

Recovery point of H.265

Table 1.630. RET_QUERY_DEC_RECOVERY_POINT 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													EXIST_FLAG	BROKEN_LINK_FLAG	EXACT_MATCH_FLAG	SIGNED_RECOVERY_POC_CNT																	
0	0	0	0	0	0	0	0	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 1.631. RET_QUERY_DEC_RECOVERY_POINT Field Description

Bit	Name	Type	Function	Reset Value
[31:19]	RSVD	R	Reserved	0x0
[18]	EXIST_FLAG	R/W	exist_flag	0x0
[17]	BROKEN_LINK_FLAG	R/W	broken_link_flag	0x0
[16]	EXACT_MATCH_FLAG	R/W	exact_match_flag	0x0
[15:0]	SIGNED_RECOVERY_POC_CNT	R/W	signed recovery_poc_cnt	0x0

1.2.5.2.11.29. RET_QUERY_DEC_DEBUG_INDEX (0x0000016C)

Table 1.632. RET_QUERY_DEC_DEBUG_INDEX 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

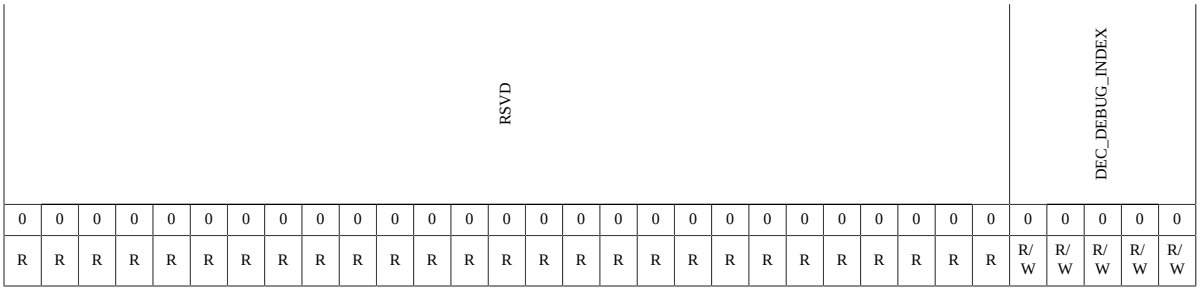


Table 1.633. RET_QUERY_DEC_DEBUG_INDEX Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	DEC_DEBUG_INDEX	R/W	Decoded picture index (internal use only)	0x0

1.2.5.2.11.30. RET_QUERY_DEC_DECODED_INDEX (0x00000170)

Decoded picture index of DPB

Table 1.634. RET_QUERY_DEC_DECODED_INDEX Bit Assignment

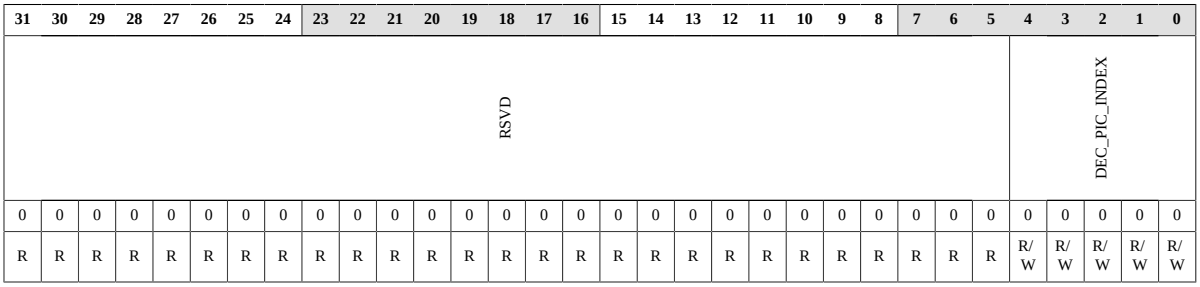


Table 1.635. RET_QUERY_DEC_DECODED_INDEX Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	DEC_PIC_INDEX	R/W	Decoded Picture Index for FBC buffer	0x0

1.2.5.2.11.31. RET_QUERY_DEC_DISPLAY_INDEX (0x00000174)

Display picture index of DPB

Table 1.636. RET_QUERY_DEC_DISPLAY_INDEX Bit Assignment

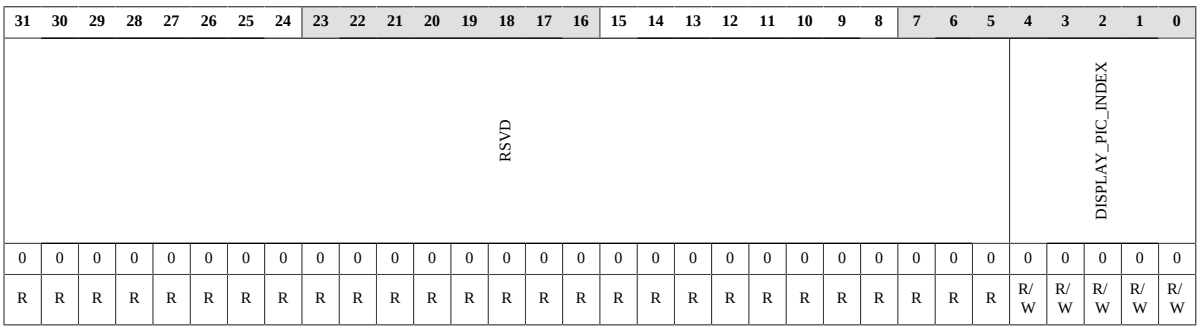


Table 1.637. RET_QUERY_DEC_DISPLAY_INDEX Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	DISPLAY_PIC_INDEX	R/W	Display Picture Index for FBC buffer (by reordering)	0x0

1.2.5.2.11.32. RET_QUERY_DEC_REALLOC_INDEX (0x00000178)

Display picture index of DPB

Table 1.638. RET_QUERY_DEC_REALLOC_INDEX 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																											DISPLAY_PIC_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	R	R	R	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 1.639. RET_QUERY_DEC_REALLOC_INDEX Field Description

Bit	Name	Type	Function	Reset Value
[31:5]	RSVD	R	Reserved	0x0
[4:0]	DISPLAY_PIC_INDEX	R/W	Display Picture Index for FBC buffer (by reordering)	0x0

1.2.5.2.11.33. RET_QUERY_DEC_DISP_IDC (0x0000017C)

Table 1.640. RET_QUERY_DEC_DISP_IDC 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
DISPLAY_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 1.641. RET_QUERY_DEC_DISP_IDC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	DISPLAY_FLAG	R/W	Display frame buffer flag.	0x0

1.2.5.2.11.34. RET_QUERY_DEC_NUM_ERR_CTB (0x00000180)

Number of error CTU

Table 1.642. RET_QUERY_DEC_NUM_ERR_CTB 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

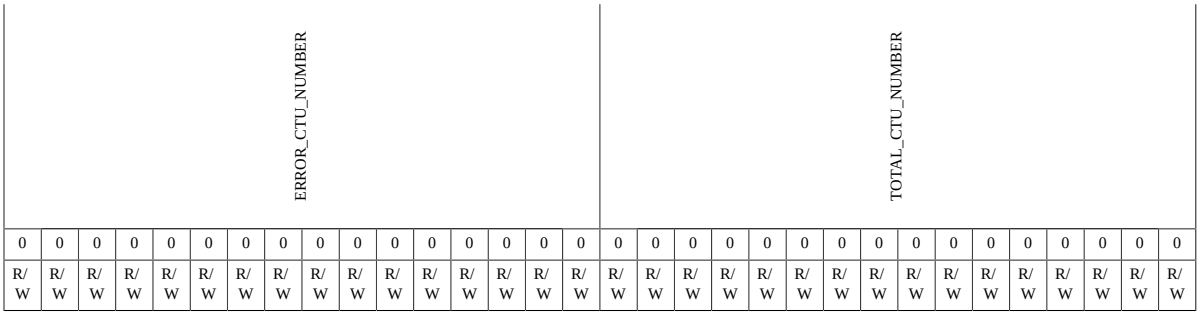


Table 1.643. RET_QUERY_DEC_NUM_ERR_CTB Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	ERROR_CTU_NUMBER	R/W	Error number of CTU in H.265 or SB of VP9.	0x0
[15:0]	TOTAL_CTU_NUMBER	R/W	Total number of CTU in H.265 or SB of VP9.	0x0

1.2.5.2.11.35. RET_QUERY_DEC_SEEK_CYCLE (0x000001D4)

Table 1.644. RET_QUERY_DEC_SEEK_CYCLE Bit Assignment

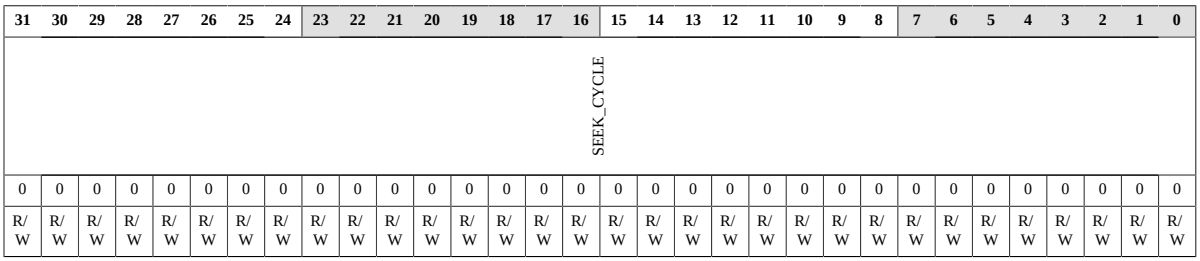


Table 1.645. RET_QUERY_DEC_SEEK_CYCLE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SEEK_CYCLE	R/W	Cycle of seeking slices of a picture	0x0

1.2.5.2.11.36. RET_QUERY_DEC_PARSING_CYCLE (0x000001D8)

Table 1.646. RET_QUERY_DEC_PARSING_CYCLE Bit Assignment

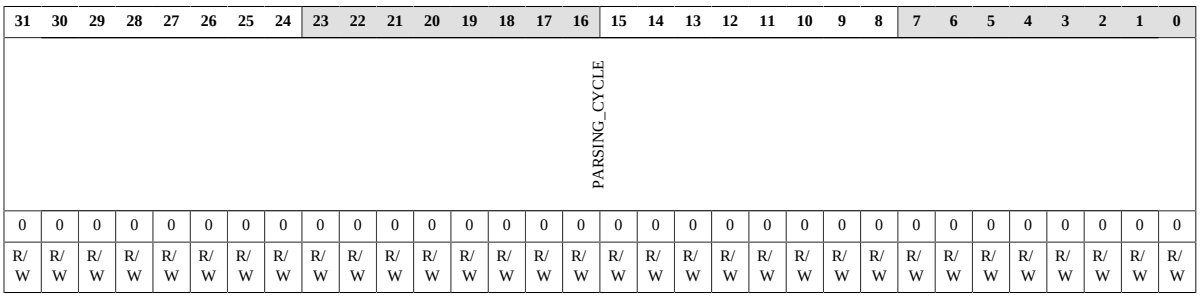


Table 1.647. RET_QUERY_DEC_PARSING_CYCLE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	PARSING_CYCLE	R/W	Cycle of parsing slices of a picture	0x0

1.2.5.2.11.37. RET_QUERY_DEC_DECODING_CYCLE (0x000001DC)

Table 1.648. RET_QUERY_DEC_DECODING_CYCLE Bit Assignment

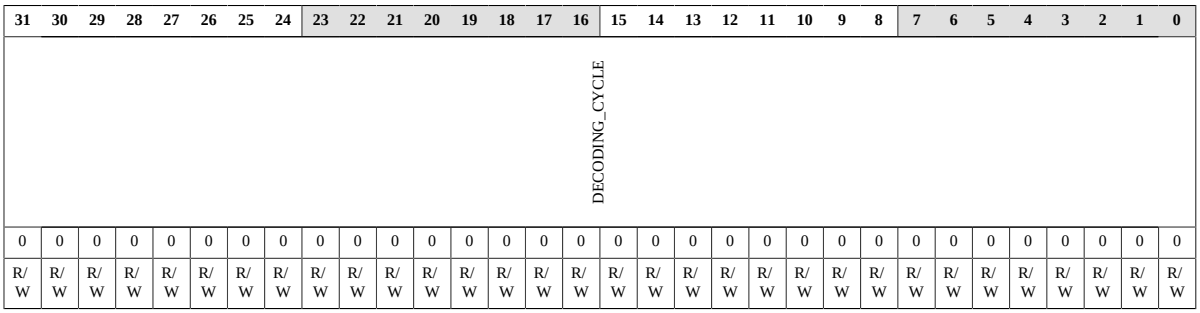


Table 1.649. RET_QUERY_DEC_DECODING_CYCLE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	DECODING_CYCLE	R/W	Cycle of decoding slices of a picture	0x0

1.2.5.2.11.38. RET_QUERY_DEC_FRAME_CYCLE (0x000001E0)

Table 1.650. RET_QUERY_DEC_FRAME_CYCLE Bit Assignment

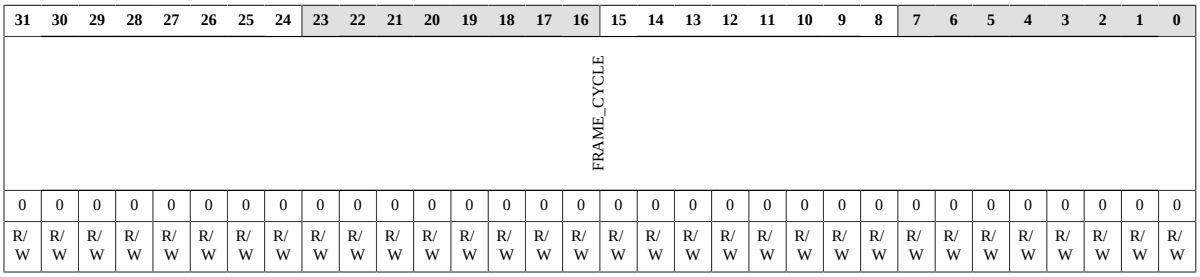


Table 1.651. RET_QUERY_DEC_FRAME_CYCLE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FRAME_CYCLE	R/W	Cycle of reconstructing a picture	0x0

1.2.5.2.11.39. RET_DEC_WARN_INFO (0x000001E4)

Warning Information

Table 1.652. RET_DEC_WARN_INFO Bit Assignment

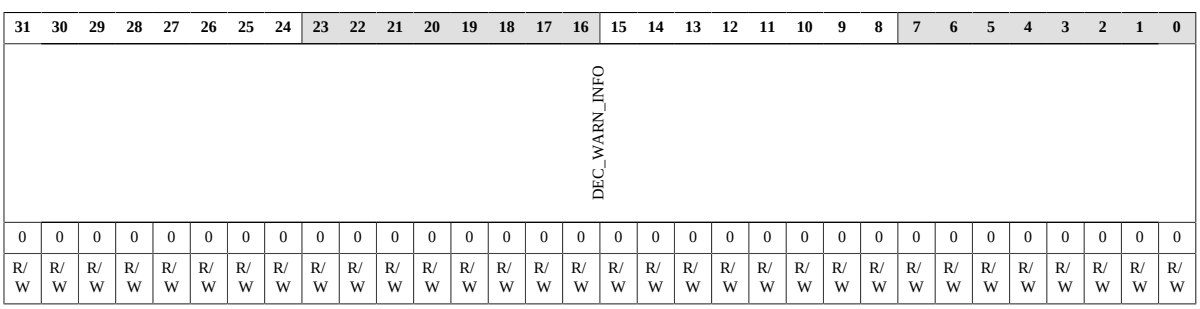


Table 1.653. RET_DEC_WARN_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	DEC_WARN_INFO	R/W	Please refer to *Section C.3, "Decode Warning Information"* defining warnings that VPU might have.	0x0

1.2.5.2.11.40. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.654. RET_BS_EMPTY Bit Assignment

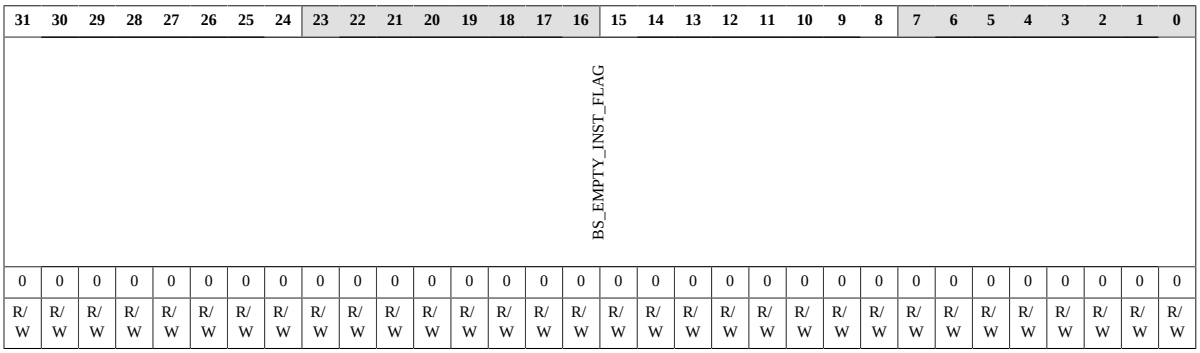


Table 1.655. RET_BS_EMPTY Field Description

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.	0x0

1.2.5.2.11.41. RET_DEC_ERR_INFO (0x000001E8)

Error Information

Table 1.656. RET_DEC_ERR_INFO Bit Assignment

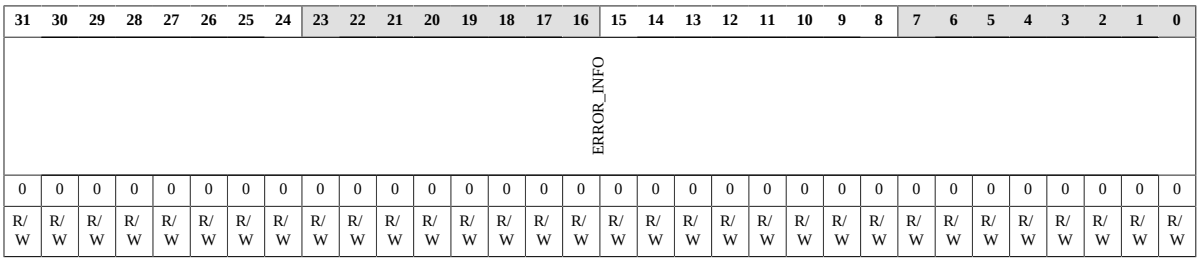


Table 1.657. RET_DEC_ERR_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	ERROR_INFO	R/W	Please refer to *Section C.2, "Decode Error Information"* defining errors that VPU might have.	0x0

1.2.5.2.11.42. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.658. RET_QUEUED_CMD_DONE Bit Assignment

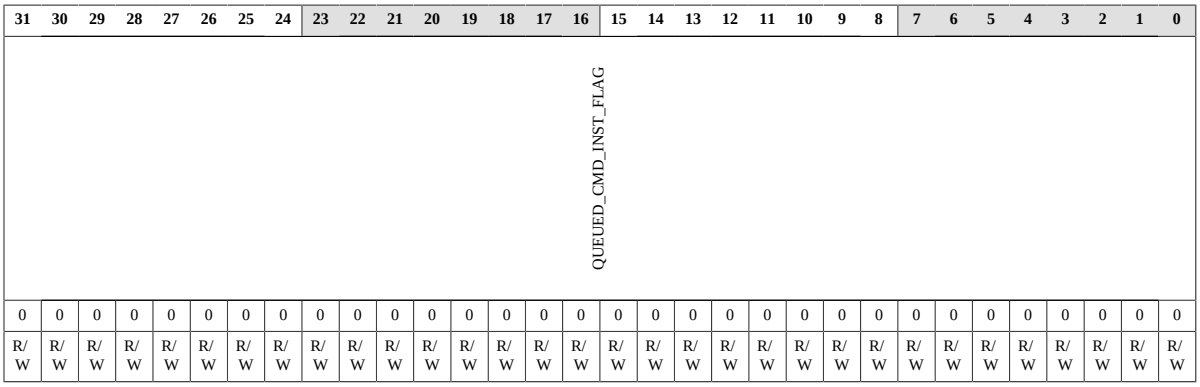


Table 1.659. RET_QUEUED_CMD_DONE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	QUEUED_CMD_INST_FLAG	R/W	QUEUED_CMD_INST_FLAG = 1 << instance_index When VPU completes its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.11.43. RET_QUERY_DEC_SUCCESS (0x000001EC)

Table 1.660. RET_QUERY_DEC_SUCCESS Bit Assignment

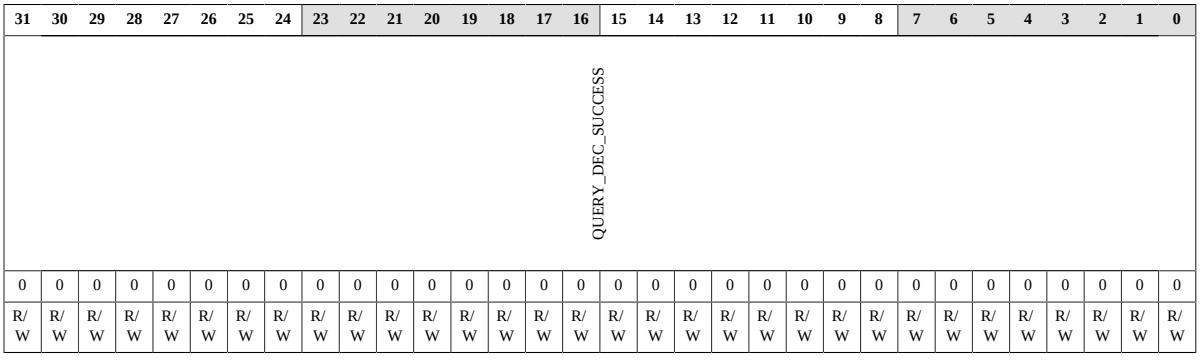
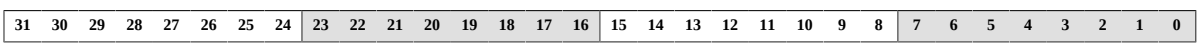


Table 1.661. RET_QUERY_DEC_SUCCESS Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	QUERY_DEC_SUCCESS	R/W	It indicates that decoding result has been successfully reported by QUERY command .	0x0

1.2.5.2.11.44. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.662. RET_SEEK_INSTANCE_INFO Bit Assignment



RSVD																								SEEK_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.663. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.11.45. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.664. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.665. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.11.46. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.666. 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

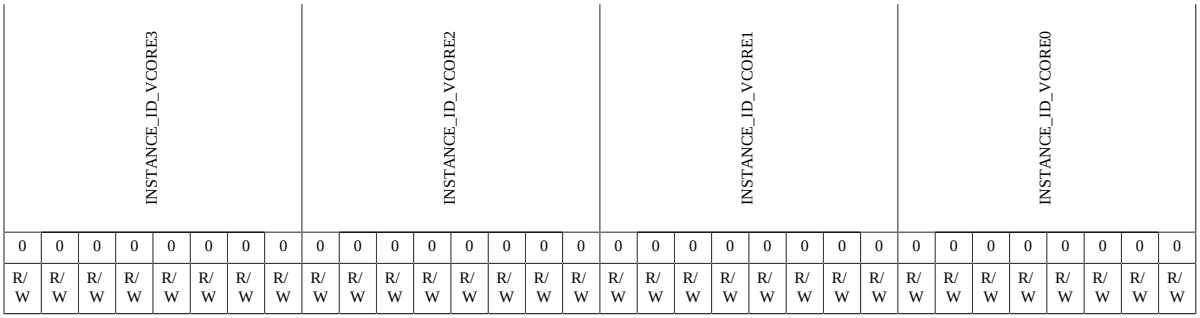


Table 1.667. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.11.47. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.668. RET_DONE_INSTANCE_INFO Bit Assignment

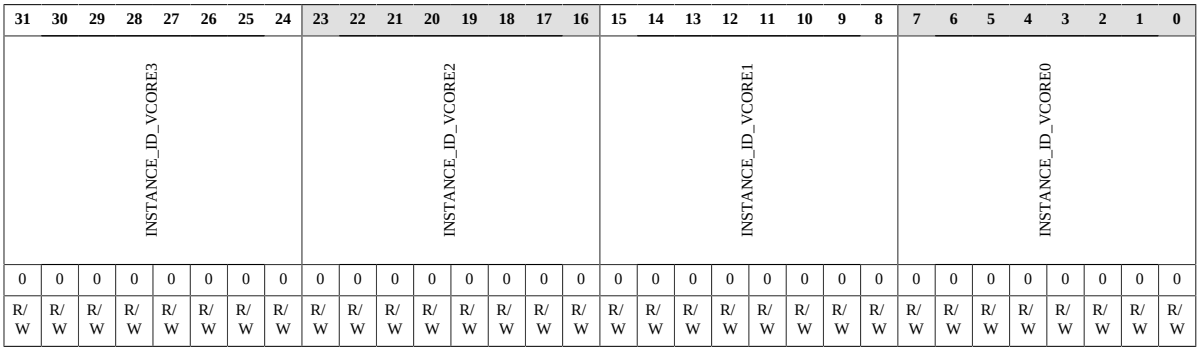


Table 1.669. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.12. QUERY(UPDATE_DISP_IDC) Command Parameter Registers

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

1.2.5.2.12.1. 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 **QUERY** command (**0x4000** is given to this register).

Table 1.670. 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.671. COMMAND Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	COMMAND	R/W	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 : DEC_PIC 0x4000 : **QUERY** 0x8000 : UPDATE_BS	0x0

1.2.5.2.12.2. CMD_QUERY_OPTION (0x00000104)

Table 1.672. 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 1.673. 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 0x01: SET_WRITE_PROTECTION	0x0

Bit	Name	Type	Function	Reset Value
			0x02: GET_RESULT 0x03: UPDATE_DISP_IDC	

1.2.5.2.12.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.674. RET_SUCCESS Bit Assignment

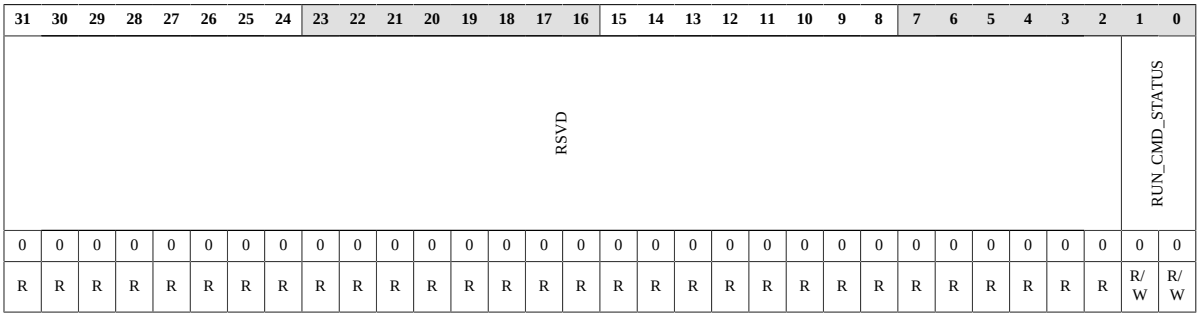


Table 1.675. RET_SUCCESS Field Description

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_WARNIG	0x0

1.2.5.2.12.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.676. RET_FAIL_REASON Bit Assignment

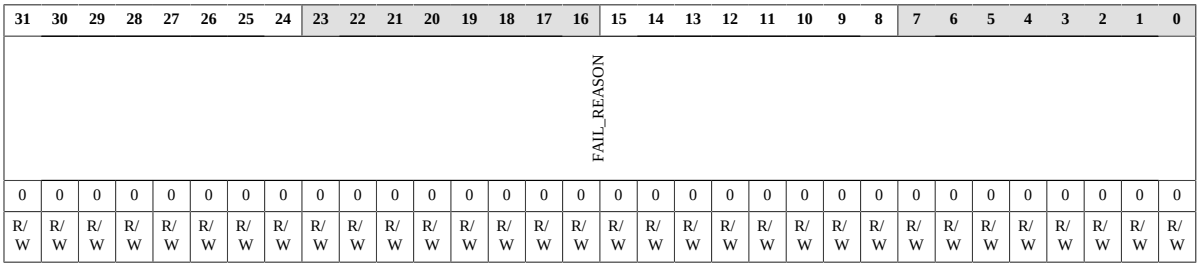
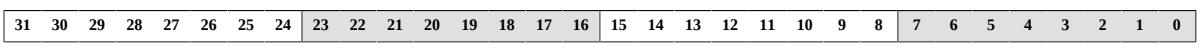


Table 1.677. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.12.5. CMD_INSTANCE_INFO (0x00000110)

Table 1.678. CMD_INSTANCE_INFO Bit Assignment



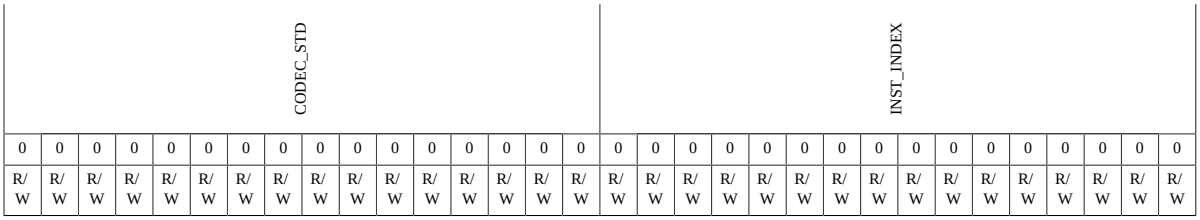


Table 1.679. CMD_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:16]	CODEC_STD	R/W	Codec standard to run HEVC_DEC = 0x0 VP9_DEC = 0x16	0x0
[15:0]	INST_INDEX	R/W	Instance Index VPU can decode more than one decoding instance simultaneously. If multiple instances are running, each instance must have their 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 65535, inclusive.	0x0

1.2.5.2.12.6. CMD_QUERY_DEC_SET_DISP_IDC (0x00000118)

Table 1.680. CMD_QUERY_DEC_SET_DISP_IDC Bit Assignment

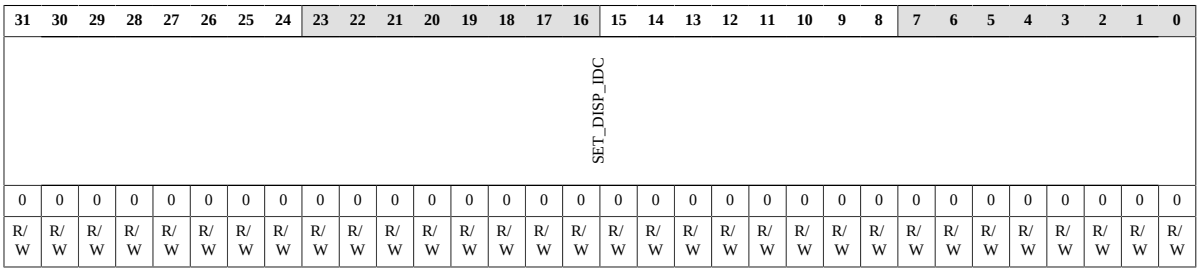
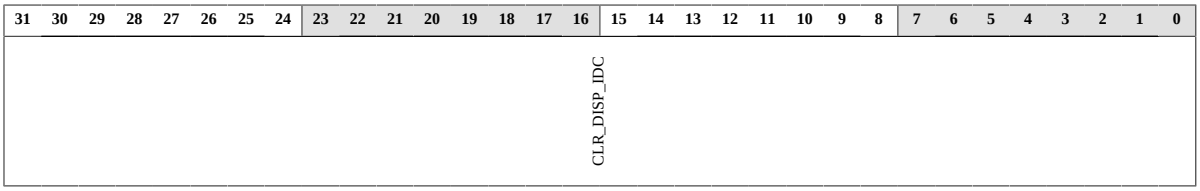


Table 1.681. CMD_QUERY_DEC_SET_DISP_IDC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	SET_DISP_IDC	R/W	Set display flag value. VPU and Host handshake display frame buffer ownership. For example, display flag for frame buffer 0 is 1, host have ownership for the frame buffer 0 and VPU can't write any data to frame buffer 0 until host release and set 0 to display flag of frame buffer 0. SET_DISP_IDC and CLR_DISP_IDC is used to set/clear display flag for 0~31 display frame buffer.	0x0

1.2.5.2.12.7. CMD_QUERY_DEC_CLR_DISP_IDC (0x0000011C)

Table 1.682. CMD_QUERY_DEC_CLR_DISP_IDC 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 1.683. CMD_QUERY_DEC_CLR_DISP_IDC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	CLR_DISP_IDC	R/W	Clear display flag value	0x0

1.2.5.2.12.8. RET_QUERY_DEC_DISP_IDC (0x0000017C)

Table 1.684. RET_QUERY_DEC_DISP_IDC 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_DISP_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 1.685. RET_QUERY_DEC_DISP_IDC Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	DEC_DISP_IDC	R/W	display frame buffer indication	0x0

1.2.5.2.12.9. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.686. 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 1.687. 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.	0x0

Bit	Name	Type	Function	Reset Value
			This field is ignored after BS_EMPTY interrupt service is done.	

1.2.5.2.12.10. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.688. 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 1.689. 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 its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	0x0

1.2.5.2.12.11. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.690. 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_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.691. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.12.12. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.692. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.693. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.12.13. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.694. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.695. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0

Bit	Name	Type	Function	Reset Value
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.12.14. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.696. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.697. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	0x0

1.2.5.2.13. UPDATE_BS Command Parameter Registers

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

1.2.5.2.13.1. 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 **UPDATE_BS** command (**0x8000** is given to this register).

Table 1.698. 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 1.699. 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 : DEC_PIC 0x4000 : QUERY 0x8000 : **UPDATE_BS**	0x0

1.2.5.2.13.2. CMD_UPDATE_BS_OPTION (0x00000104)

DEC_PIC command option

Table 1.700. 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 1.701. CMD_UPDATE_BS_OPTION Field Description

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

1.2.5.2.13.3. RET_SUCCESS (0x00000108)

Result of the run command

Table 1.702. 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 1.703. 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_WARNIG	0x0

1.2.5.2.13.4. RET_FAIL_REASON (0x0000010C)

Fail reason of the run command

Table 1.704. 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 1.705. RET_FAIL_REASON Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	FAIL_REASON	R/W	Please refer to *Section C.1, "System Error"* that lists the error types that VPU might have.	0x0

1.2.5.2.13.5. CMD_BS_WR_PTR (0x0000011C)

Bistream buffer write pointer

Table 1.706. CMD_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
WR_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.707. CMD_BS_WR_PTR Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	WR_PTR	R/W	End address of bitstream for handling current command Host can update this register anytime. If a bitstream_empty interrupt is asserted, host processor should do either feed more bitstream and update WR_PTR or set EXPLICT_END to complete decoding anyhow.	0x0

1.2.5.2.13.6. CMD_BS_OPTIONS (0x00000120)

Bistream buffer option

Table 1.708. CMD_BS_OPTIONS 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																														STREAM_END	EXPLICIT_END
0	0	0	0	0	0	0	0	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 1.709. CMD_BS_OPTIONS Field Description

Bit	Name	Type	Function	Reset Value
[31:2]	RSVD	R/W	Reserved	0x0
[1]	STREAM_END	R/W	This field makes VPU assume Stream End when there is no stream to feed in the buffer.	0x0
[0]	EXPLICIT_END	R/W	Explicit End When this field is set to 1, VPU assumes that bitstream buffer has at least one single frame and follows the tasks below. 1. VPU decodes until the end of bitstream buffer (WR_PTR) even if the last 3 bytes are all 0. 2. VPU returns success if it has decoded a frame successfully. If bitstream is insufficient to complete decoding a frame, VPU performs what it is supposed to do with the specified task in BS_SHORTAGE_OPTION of BS_PARAM. If this flag is 0, 1. VPU decodes to the almost end of bitstream buffer (WR_PTR), but not to some bytes (less than 3). It intentionally does not consume some a few bytes, because VPU is not sure if the last bytes are the start code of next NAL or they might not fill the offset in the CABAC. 2. VPU returns success if it has decoded a frame successfully. If bitstream is insufficient to complete decoding a frame, VPU stops decoding and waits for more bitstream to be filled. Then host processor can do either feed bitstream more or set EXPLICIT_END to complete decoding anyhow. BITSTREAM_EMPTY interrupt is asserted when EXPLITCT_END = 0 or when bitstream buffer is near empty (not real empty) for seamless decoding.	0x0

Bit	Name	Type	Function	Reset Value
			Caution Host processor can set this register any time, but cannot clear during command processing.	

1.2.5.2.13.7. RET_BS_EMPTY (0x000001E4)

Reserved for CPB empty interrupt

Table 1.710. RET_BS_EMPTY Bit Assignment

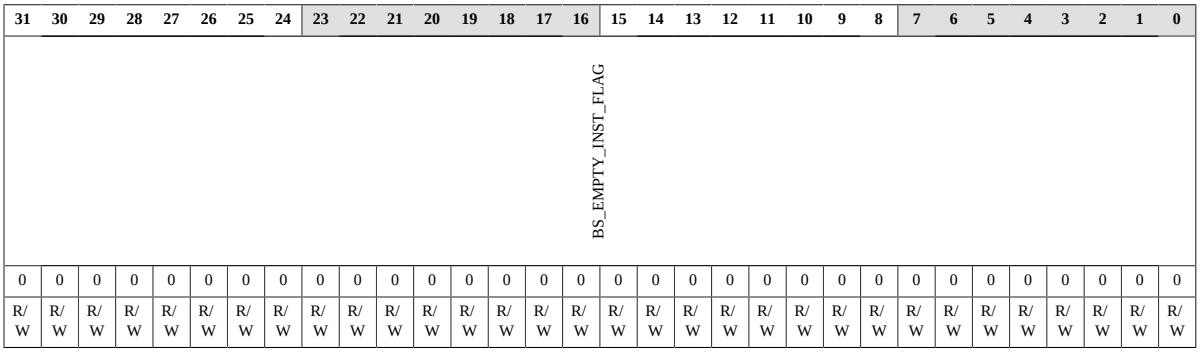


Table 1.711. RET_BS_EMPTY Field Description

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.	0x0

1.2.5.2.13.8. RET_QUEUED_CMD_DONE (0x000001E8)

Table 1.712. RET_QUEUED_CMD_DONE Bit Assignment

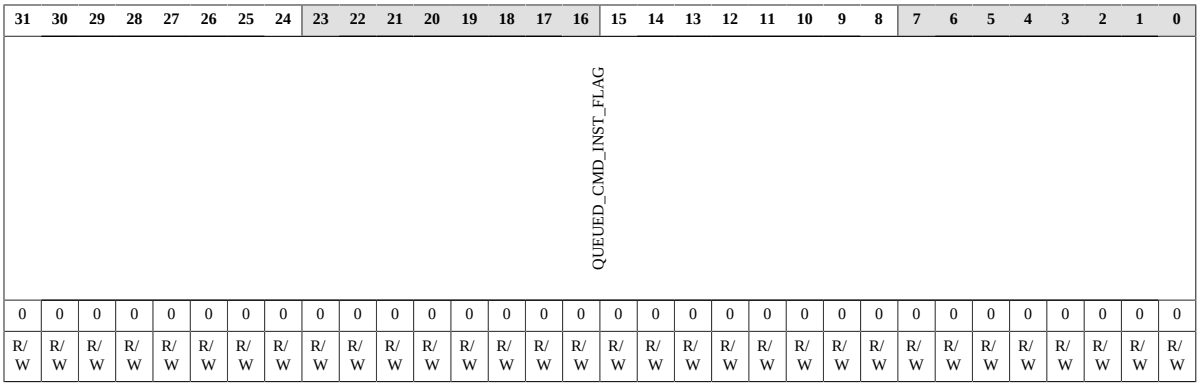


Table 1.713. RET_QUEUED_CMD_DONE Field Description

Bit	Name	Type	Function	Reset Value
[31:0]	QUEUED_CMD_INST_FLAG	R/W	QUEUED_CMD_INST_FLAG = 1 << instance_index.	0x0

Bit	Name	Type	Function	Reset Value
			When VPU completes its internal processing for INIT_SEQ command or DEC_PIC command of an instance and is ready to output the processing result to host, VPU triggers an interrupt to host with the instance index. Host can recieve output result information of the instance indicated by intance_index. This field is ignored after INIT_SEQ or DEC_PIC interrupt service is done.	

1.2.5.2.13.9. RET_SEEK_INSTANCE_INFO (0x000001EC)

Table 1.714. 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_INSTANCE_INFO							
0	0	0	0	0	0	0	0	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 1.715. RET_SEEK_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:8]	RSVD	R	Reserved	0x0
[7:0]	SEEK_INSTANCE_INFO	R/W	The instance ID in seeking phase which runs on on V-CPU	0x0

1.2.5.2.13.10. RET_PARSING_INSTANCE_INFO (0x000001F0)

Table 1.716. 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
INSTANCE_ID_PRESCAN_CORE3								INSTANCE_ID_PRESCAN_CORE2								INSTANCE_ID_PRESCAN_CORE1								INSTANCE_ID_PRESCAN_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/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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 1.717. RET_PARSING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_PRESCAN_CORE3	R/W	The instance ID in parsing phase which runs on V-CORE #3	0x0

Bit	Name	Type	Function	Reset Value
[23:16]	INSTANCE_ID_PRESCAN_CORE2	R/W	The instance ID in parsing phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_PRESCAN_CORE1	R/W	The instance ID in parsing phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_PRESCAN_CORE0	R/W	The instance ID in parsing phase which runs on V-CORE #0	0x0

1.2.5.2.13.11. RET_DECODING_INSTANCE_INFO (0x000001F4)

Table 1.718. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.719. RET_DECODING_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID in decoding phase which runs on V-CORE #3	0x0
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID in decoding phase which runs on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID in decoding phase which runs on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID in decoding phase which runs on V-CORE #0	0x0

1.2.5.2.13.12. RET_DONE_INSTANCE_INFO (0x000001FC)

Table 1.720. 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
INSTANCE_ID_VCORE3								INSTANCE_ID_VCORE2								INSTANCE_ID_VCORE1								INSTANCE_ID_VCORE0							
0	0	0	0	0	0	0	0	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 1.721. RET_DONE_INSTANCE_INFO Field Description

Bit	Name	Type	Function	Reset Value
[31:24]	INSTANCE_ID_VCORE3	R/W	The instance ID which has been done on V-CORE #3	0x0

Bit	Name	Type	Function	Reset Value
[23:16]	INSTANCE_ID_VCORE2	R/W	The instance ID which has been done on V-CORE #2	0x0
[15:8]	INSTANCE_ID_VCORE1	R/W	The instance ID which has been done on V-CORE #1	0x0
[7:0]	INSTANCE_ID_VCORE0	R/W	The instance ID which has been done on V-CORE #0	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"*.

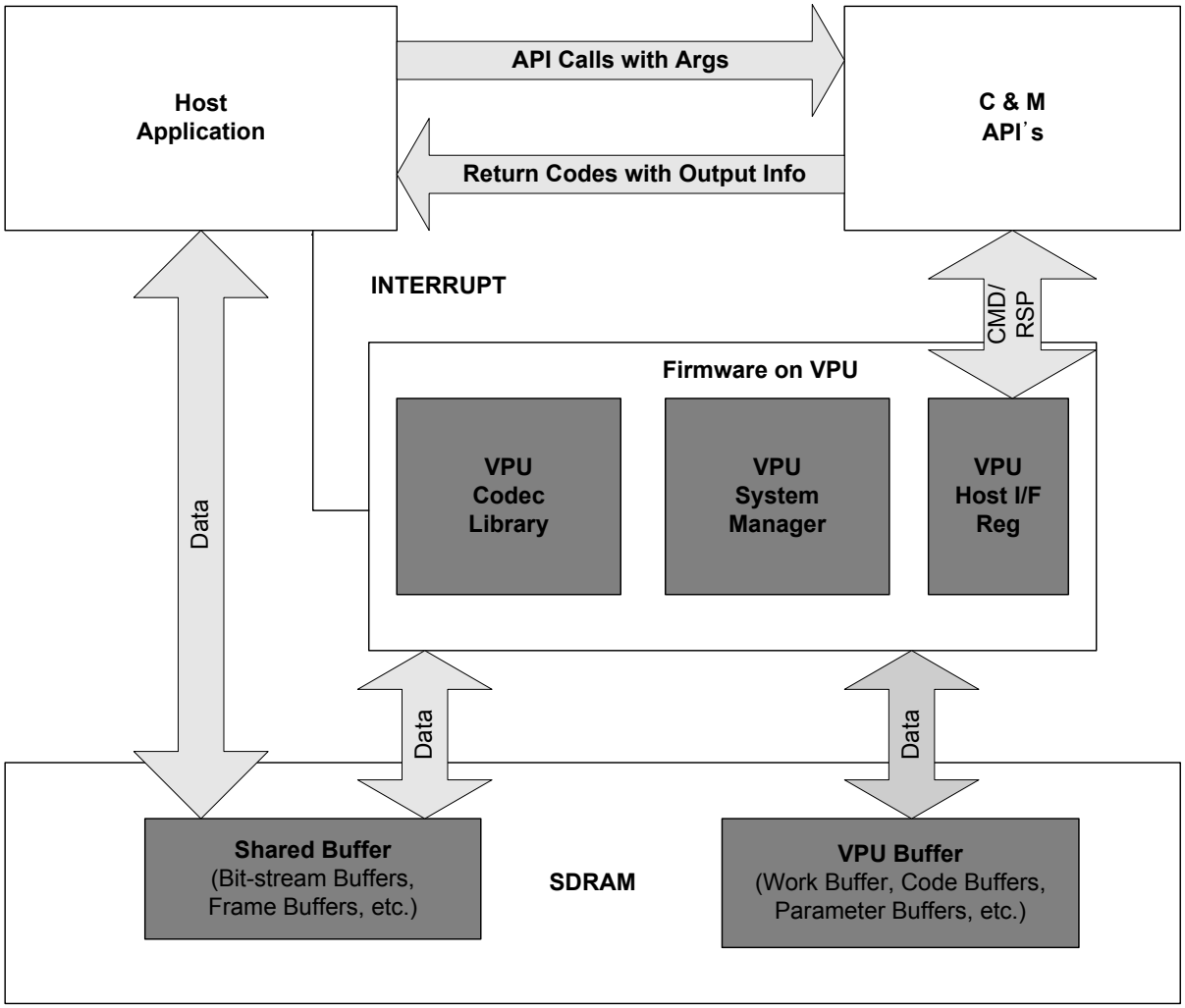


Figure 2.1. SW control model of VPU from host application

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.

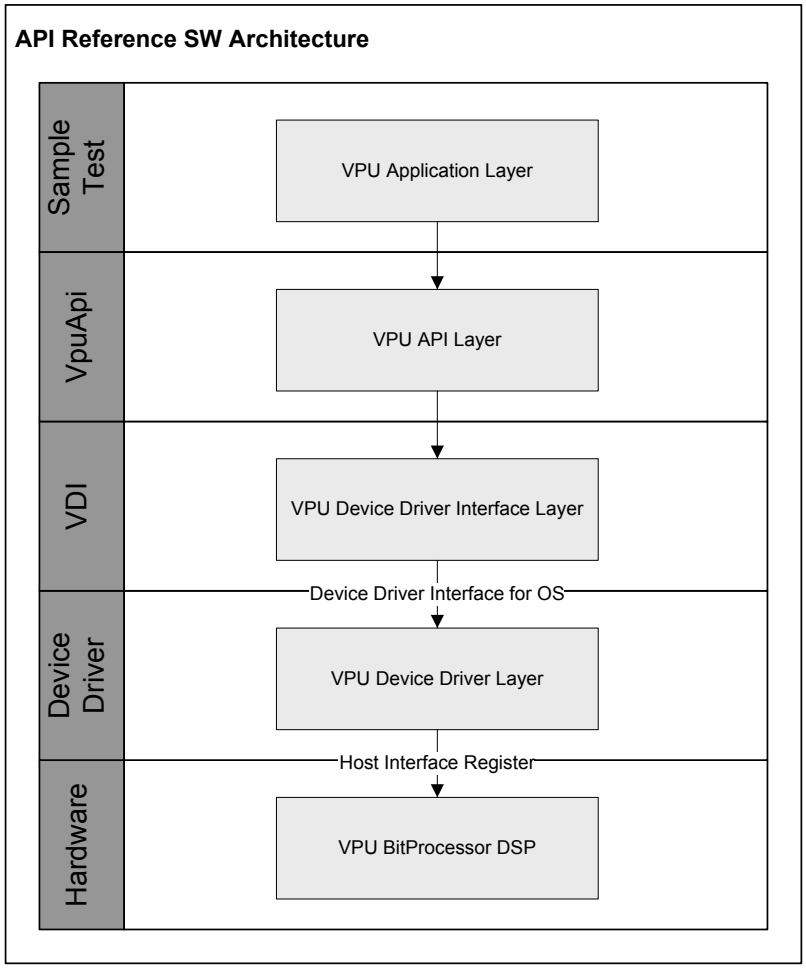


Figure 2.2. API Reference Software Architecture

- 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, Windows, 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>/

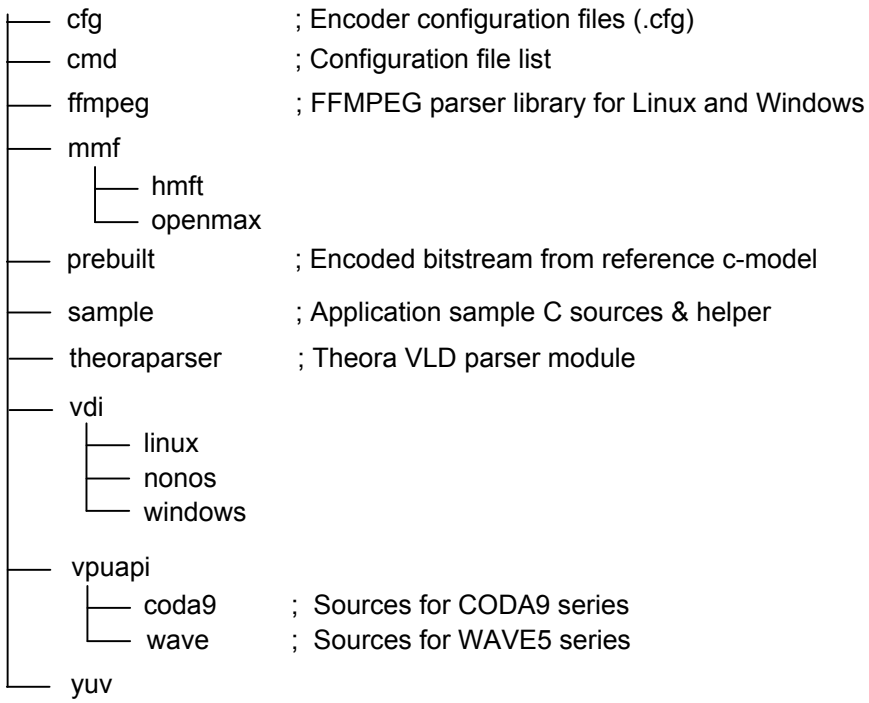


Figure 2.3. Source Tree of VPU API Reference Software

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.

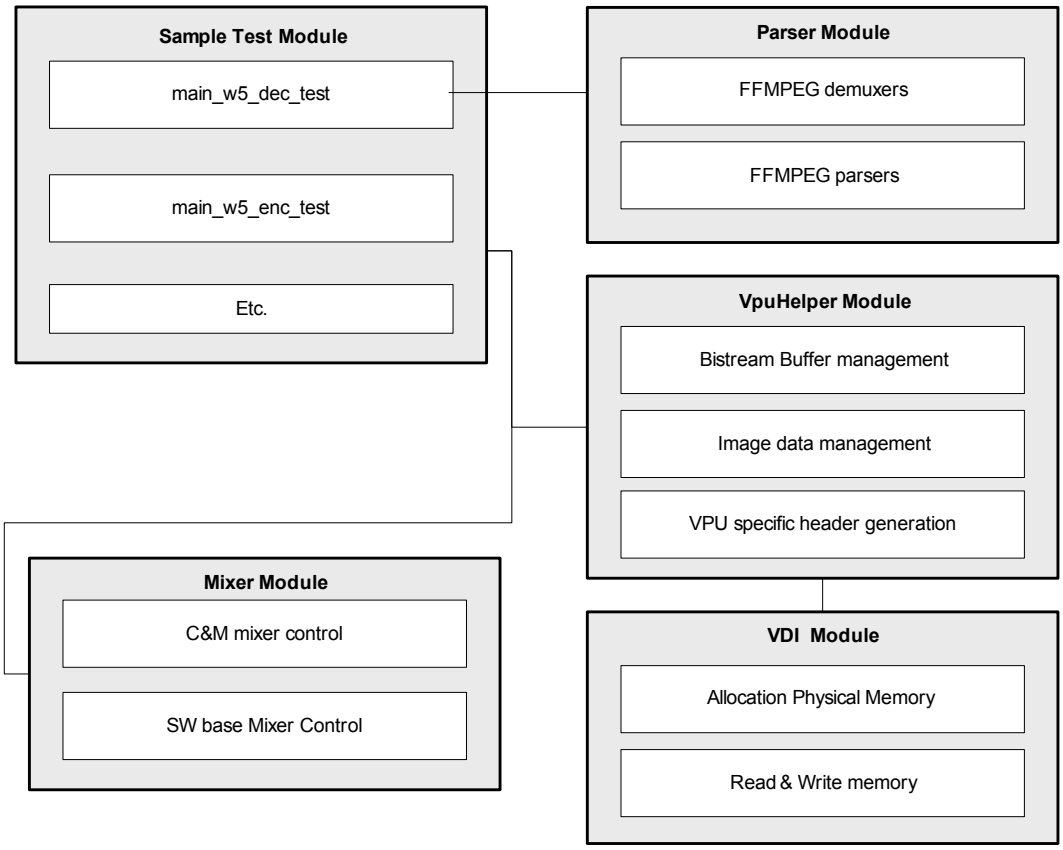


Figure 2.4. Application Layer

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.

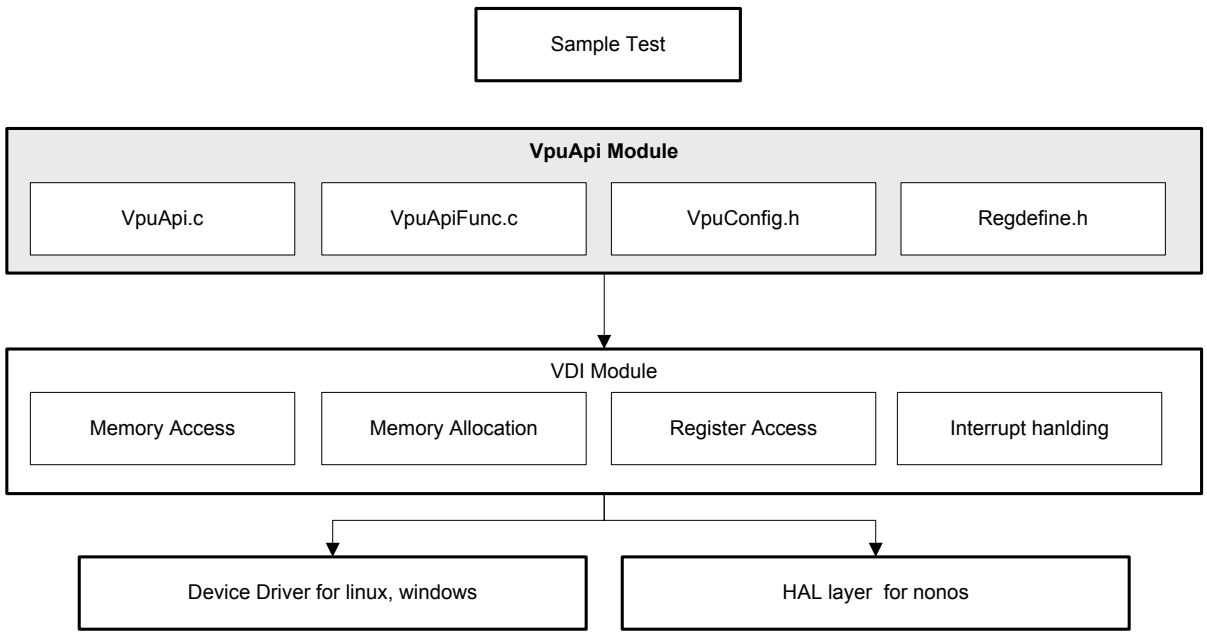


Figure 2.5. VPU API Layer

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, Windows CE, Windows and so forth, 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.

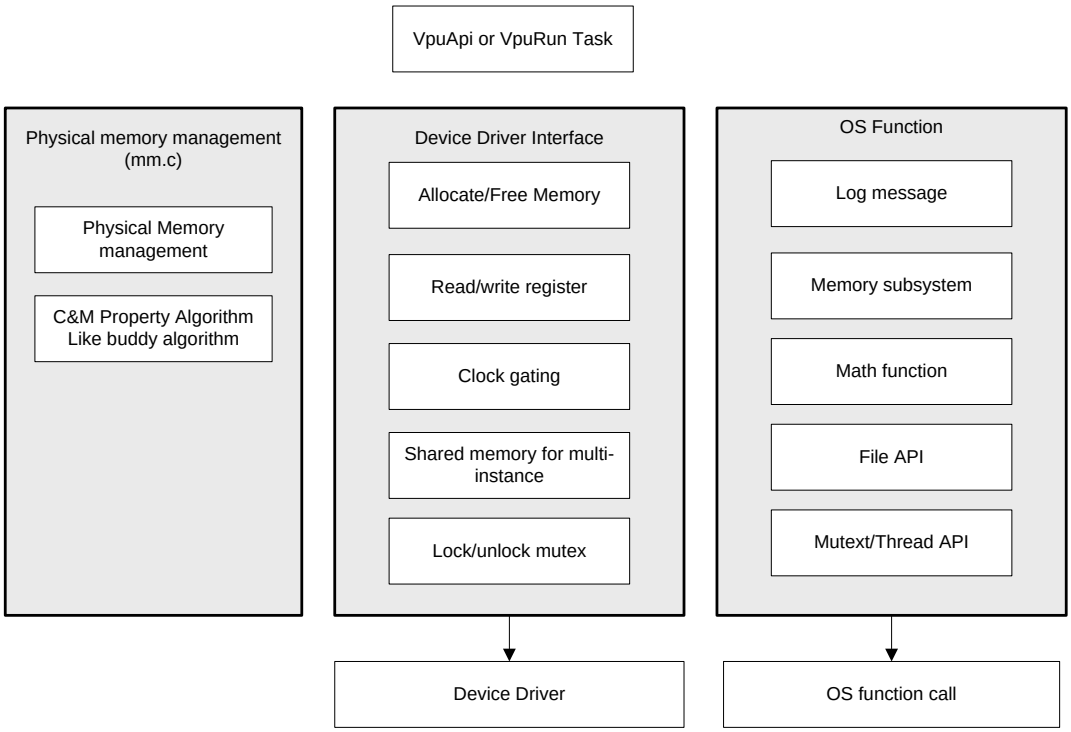


Figure 2.6. VDI Layer

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

2.2.3.4. Windows8

VPU Reference Software is ported to Windows7 and Window8, with including HMFT Plugin and AVStream Class driver or WDDM device driver for DXVA. The relevant codes are in VDI/windows folder.

- Build Environment
  - Visual Studio 2010, 2012
  - Windows Driver kit to build kernel mode driver
- Verification Platform
  - Windows7/8 installed PC and Chips&Media FPGA board which is connected to PCI card.

2.2.3.5. Chips&Media FPGA

Chips&Media FPGA is ported to Linux OS with VDI codes in VDI/linux folder. The pre-processor CNM_FPGA_PLATFORM should be indicated in project file.

- Build Env
  - Linux
  - Wave5xxDec/Enc.mak (or Wave5xx_multi.mak)
- Verification Platform
  - Chips&Media FPGA + Linux OS for HOST
  - Connect the two ones with PCI card. Protocol is like SRAM interface.
  - A memory for video resides on Chips&Media FPGA board.

2.3. Porting to Target System

2.3.1. Identifying Build Tool(c - compiler)

Example 2.1. config.h

```
--------------------------------------------------------------------------------
'#if defined(_WIN32) || defined(__WIN32__) || defined(_WIN64) || defined(WIN32) ||
  defined(__MINGW32__)
'#define PLATFORM_WIN32
'#elif defined(linux) || defined(__linux) || defined(ANDROID)
'#define PLATFORM_LINUX
```

```
'#else
'#define PLATFORM_NON_OS
'#endif

'#if defined(_MSC_VER)
'#include <windows.h>
'#define inline _inline
'#elif defined(__GNUC__)
'#elif defined(__ARMCC__)
'#else
'#error "Unknown compiler."
'#endif
--------------------------------------------------------------------------------
```

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_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_WAVE512_SRAM_SIZE 0x2E000

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.

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.

WAVE512DEC_WORKBUF_SIZE/WAVE520ENC_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 or Windows CE, 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 both values of ENDIAN and AXIID.)
   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. Write VPU_BUSY_STATUS register to 1.
11. Set COMMAND(0x100) register to 1 for INIT_VPU command.
12. 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 subsections.

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 DEC_PIC_HDR 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 indiates 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:

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. Decoder Control

3.2.1. Overall Decoder Sequence

Figure 3.1, "Decoder Control Flow with APIs" gives a brief summary of decoder control flow that shows the relation between decoder control flow and API function. And it also represents short description of each stage.

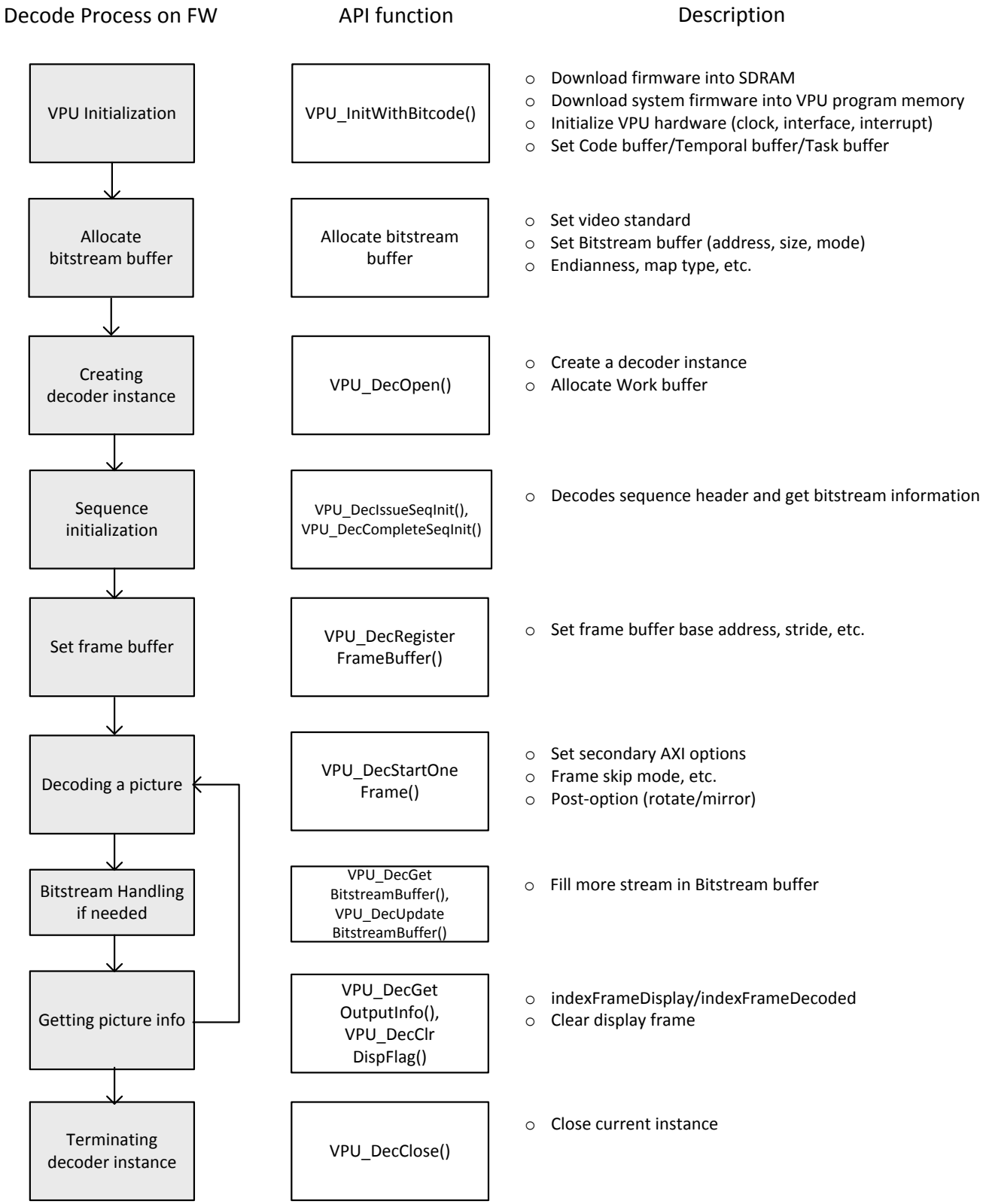


Figure 3.1. Decoder Control Flow with APIs

3.2.2. Creating a Decoder Instance

After initialization of VPU, the first step to run decoder operation should be creation of a decoder instance, and acquisition of a handle that specifies the created decoder instance. It could be easily done by using a single API function called `VPU_DecOpen()`.

When creating a new decoder instance, host application should specify the internal features of this decoder instance through `DecOpenParam` structure. This structure includes the following information about the new decoder instance:

- Bitstream buffer address & size

  Physical address of bitstream buffer start address and its size

- Codec standard

  Video decoder standard, H.265/HEVC

3.2.3. Configuring VPU for Decoder Instance

3.2.3.1. Feeding Bitstream into Bitstream Buffer

In case of decoder, sequence initialization `INIT_SEQ` performs parsing of high level header syntaxes such as SPS/PPS in H.265/HEVC to get sequence information. So host application needs to fill the bitstream buffer with enough bitstream ahead of `INIT_SEQ`.

In some applications, they cannot guarantee that these kinds of header syntaxes are always found at the beginning of bitstream. In this case, host application should keep on feeding input data stream to bitstream buffer until VPU successfully gets all the required information from input data stream.

In ring-buffer mode, host application should know available space in bitstream buffer to feed input bitstream. It could be easily determined by using a read pointer, a write pointer, and a bitstream buffer size. Getting available space in bitstream buffer, host application can directly download decoder input stream to bitstream buffer. After finishing stream download, host application should inform the amount of downloaded stream by updating the stream write pointer.

The VPU API provides an updated API function to get a read pointer, a write pointer and available space by one function call, `VPU_DecGetBitstreamBuffer()`. And updating a write pointer could also be done easily by using an API function, `VPU_DecUpdateBitstreamBuffer()`.

Note For better understanding of this operation, you can refer to WriteBsBufHelper() function in src/vpuhelper.c file.

3.2.3.2. Sequence Initialization

After creating a new instance and feeding input bitstream to the bitstream buffer, host application can give INIT_SEQ command to VPU in order to get sequence information from bitstream. After parsing header syntaxes, VPU returns crucial information for decoder configuration including:

- Picture size

  Picture width and height

- Picture cropping rectangle information (also known as conformance window)

  Information about H.265/HEVC decoder picture cropping rectangle which presents the offset of top-left point and bottom-right point from the origin of frame buffer.

- Frame rate

  Decoder frame rate

- Minimum number of Frame Buffers

  The number of frame buffers to be required for VPU to save reconstructed frames

- Frame buffer delay for display reordering

  The number of frame delays for supporting display reordering in H.265/HEVC decoder

- Scaler

  For downscaling frames, scaler parameters should be given before setting framebuffer information. The size of width and height to be down-scaled need to be a multiple of 8, and up to 1/8 of original size is allowed. Host application can give DEC_SET_SCALER_INFO command of VPU_DecGiveCommand() along with ScalerInfo structure pointer indicating the size information for scaler operation.

  After a frame has been decoded, calling VPU_DecGetOutputInfo() returns the scaled-down picture size and framebuffer information through dispPicWidth, dispPicHeight and dispFrame field of DecGetOutputInfo structure.

Display Reordering

Frame buffer delay is an H.265/HEVC-specific parameter for supporting display reordering. If host application decides to support display reordering and reordering requires 5 additional frame buffers, for example, then the first display output comes out from decoder after decoding 6-th frame. Theoretically, maximum 16-frame delay would happen in display reordering.

API Funtion for Sequence Init

VPU provides API functions `VPU_DecIssueSeqInit()` and `VPU_DecCompleteSeqInit()` to handle INIT_SEQ operation at an API level. Completion of this function is signaled by a dedicated interrupt to host processor. Or host processor can check by polling `BusyFlag`.

Error Handling

An important issue concerning INIT_SEQ operation is error-handling. It is because any error in high layer header syntaxes might cause severe problem in decode operation. Generally lots of marker bits are added into these header syntaxes in order to help error detection.

In case that header syntaxes included in the stream have crucial errors or header syntaxes are not received for a long time, VPU might be stuck on this task and no other instances run on VPU at all. So VPU API provides a special function which could be used in this problematic situation only, called `VPU_DecSetSeqInitEsc()`.

When this function is called and stream buffer is empty, VPU automatically terminates INIT_SEQ operation. Then host application could decide whether closing this instance, retrying INIT_SEQ after running a different decoder instance. After escaping from this situation, it is highly recommended to reset the internal ESCAPE flag by calling `VPU_DecSetSeqInitEsc()` function again.

- Thumbnail mode

Host application can set thumbnail mode for INIT_SEQ operation. In thumbnail mode, VPU can decode only I-RAP pictures with a minimum number of DPB.

3.2.3.3. Registering Frame Buffers

This configuring process is finished by registering frame buffers to VPU for picture decode operation. In this final stage of configuration, the minimum number of frame buffer, which is one of the returned parameter from

`VPU_DecCompleteSeqInit()`, has a very important meaning. This parameter means that host application should reserve at least the same number of frame buffers to VPU for proper decoding operation.

The size of the frame buffers can be calculated from picture width and height. When both picture width and picture height are multiple of 16, picture size (width x picture) is same as the frame buffer size. Also host application can apply ceiling operation to picture width and picture height to get the smallest number in multiples of 16.

The addresses of the frame buffers might not be continuous, and the address of color planes of a frame also might be discontinuous.

3.2.4. Running Picture Decode on VPU

3.2.4.1. Initiating Picture Decode

When activating picture decoding operation, host application should provide the following information to VPU:

- Trick Mode
  - Non I-RAP skip
  - Thumbnail mode It has the same effect on VPU as Non I-RAP skip, except that VPU does not handle reference frames in the thumbnail mode. It works only when INIT_SEQ with enabling thumbnail mode is executed.
  - Non Ref skip
- Frame Skip Mode

  Enable or disable skipping bitstream for the next frame decoding.
- Secondary AXI

  VPU provides an option for use of secondary AXI to save bandwidth. The secondary AXI allows VPU to save temporal data on internal SRAM instead of on external SDRAM. There are three hardware blocks which can be connected to SRAM through secondary AXI: VCE_LF_ROW_USE, VCE_LF_COL_USE, and VCE_IP_USE.

  Host processor can decide which block should be enabled for secondary AXI in consideration of system resources and set the base address of each area which used by each purpose. This process is done by calling VPU_DecGiveCommand(). If host processor skips this process, secondary AXI is disabled.

  Note Refer to the Optional SRAM Bandwidth saving excel file for detailed SRAM size that each option requires.

After providing all these parameters to VPU, host application can start picture decode operation by sending DEC_PIC command.

The VPU API provides an API for handling all these complex operations, `VPU_DecStartOneFrame()` which just initiates picture decode operation and returns the decode result. Host processor can check completion of picture decode operation as described in *Section 3.2.4.3, "Completion of Picture Decoding"*.

3.2.4.2. Decoder Stream Handling

For use of ring-buffer mode, VPU provides an API function to get a stream read pointer, a write pointer and available space by one function call, `VPU_DecGetBitstreamBuffer()`. Host application can get exact information about available space on bitstream buffer by using this API and transfer a certain amount of stream data to bitstream buffer which should be less than or equal to the available size. When transferring the stream data, host application should take care of end of bitstream buffer in order to avoid unexpected data corruption. While transferring stream data to bitstream buffer, a write pointer might become equal to the end address of bitstream buffer.

Host application should wrap around the write pointer to the beginning of bitstream buffer and go on downloading in order to avoid data corruption.

And updating a write pointer could also be done easily by using the API function, `VPU_DecUpdateBitstreamBuffer()`. Write pointer wrap-around and updating write pointer is done internally in this API function by providing just the downloaded stream size. Before updating the write pointer, host application must finish transferring stream data to bitstream buffer. If not, a certain mismatch in access time might be able to cause problems in decoder operation.

3.2.4.3. Completion of Picture Decoding

Picture decoder operation takes a certain amount of time. Host application could go on other tasks while waiting for the completion of picture decoding operation such as display processing of the previously decoded output.

Host application can use two different type of scheme for detecting completion of picture decoding operation: polling a status register or receiving an interrupt signal. If host application uses a polling scheme, it just needs to check the VPU_BUSY_STATUS register. Using the API function `VPI_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 DEC_PIC command is mapped on bit [3] of Interrupt Enable Register. So host application could easily know the completion of picture decoder operation with this dedicated interrupt signal from VPU.

3.2.4.4. Querying Decode Result

Whenever a picture decoding is completed, host application can get the decoded output such as display frame index, decoded frame index, decoded frame picture type, number of error concealed CTU, etc. The API provides `VPU_DecGetOutputInfo()` function to get the output result of picture decode operation.

- **Reading display output from VPU**

  The display frame index `indexFrameDisplay` represents a frame buffer index which is ready to be displayed. It can be different from a decoded picture index `indexFrameDecoded` for such as display reordering in H.265/HEVC.

  In the sequence initialization, there might be no display output from VPU even after several frames are actually decoded because of the display order. In H.265/HEVC reorder case, the first display output can come out after the 17th frame is decoded in worst case. Also, there might be no proper display buffer index due to frame skip option enabled. In both cases, VPU returns a negative frame buffer index. It could be -3 or -2 depending on frame skip option.

  `indexFrameDisplay` might indicate -1 when it is the end of sequence. When host application receives `indexFrameDisplay` of -1, it can terminate the current decoder instance.

  With the `indexFrameDisplay`, host application can easily figure out the status of display output as follows:

  – Non-negative value of indexFrameDisplay

    The output index value indicates the frame buffer index of display output.

  – indexFrameDisplay = -1

    It means there is no more display output when stream end is signaled to VPU. It indicates the end of sequence.

  – indexFrameDisplay = -2 or -3

    It means there is no display output temporarily due to picture reordering or skip option.

- **Reading decoded output from VPU**

VPU returns the decoded frame buffer index `indexFrameDecoded`. This index is used to represent the index of frame buffer where decoded picture is stored. In general, host application does not need to care about this index, because it . The display index, indexFrameDisplay, would be enough to handle the output of VPU decoder.

In addition, there are two more frame buffer index `indexFrameDecodedForTiled` and `indexFrameDisplayForTiled`. They are the index of frame buffer where reconstructed frame is store in compressed format by FBC(Frame Buffer Compression). When DecOpenParam::wtlEnable is false, indexFrameDecodedForTiled and indexFrameDecoded have the same index and indexFrameDisplayForTiled and indexFrameDisplay have the same index. However, when DecOpenParam::wtlEnable is true, indexFrameDecodedForTiled and indexFrameDecoded can have the different index and indexFrameDisplayForTiled and indexFrameDisplay have the different index.

Figure 3.2, "Mapping of Linear framebuffer and Compressed framebuffer when WTL enabled" shows an example of the way how Linear framebuffer and compressed framebuffer are mapped when DecOpenParam::wtlEnable is true.

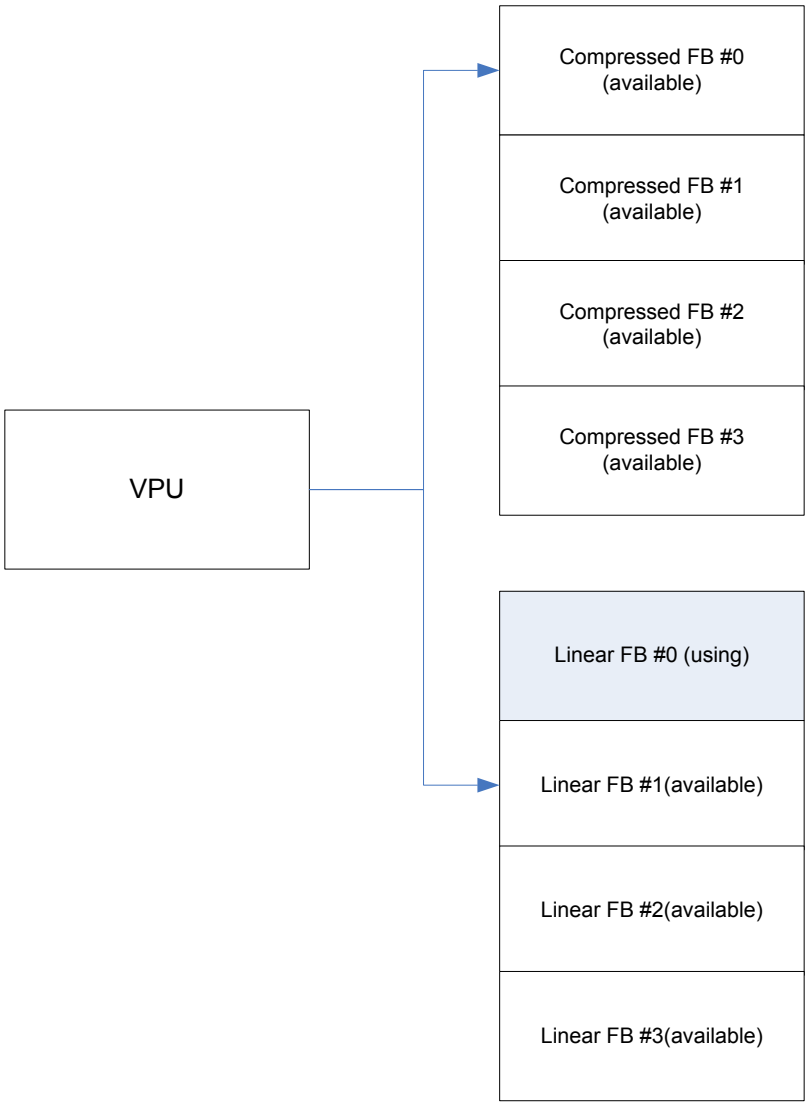


Figure 3.2. Mapping of Linear framebuffer and Compressed framebuffer when WTL enabled

Host application might need to flush out decoded frames for display. During the flushing operation, actual decoding operations is not performed. Under this situation, `indexFrameDecoded` is equal to -2 (0xFFFE) in order to represent that there is no more stream to decode at the moment. This negative decoded index is also be used when picture decoding is skipped because of skip option or picture header error.

When there is not enough frame buffer for decoded picture, this value is equal to -1 (0xFFFF). In this situation, host application needs to call `VPU_DecStartOneFrame()` again after clearing display flag with `VPU_DecClrDispFlag()`.

The following is the summary of `indexFrameDecoded`.

– Non-negative value of indexFrameDecoded

The output index value points to the frame buffer index of decoded picture.

– indexFrameDecoded == -1

The current picture has been decoded but there is no frame buffer to store the decoded picture.

– indexFrameDecoded == -2

There is no more stream to decode so that VPU is unable to continue decoding.

The following describes the relation between `indexFrameDisplay` and `indexFrameDecoded`.

Table 3.1. indexFrameDecoded and indexFrameDisplay

indexFrameDecoded	indexFrameDisplay	Description
Any value	-2 or -3	VPU decoded a picture but there is no frame for display. This is a normal case at the beginning of stream decoding due to display ordering (delay).
Any value	Any value	Normal decoding
-2	Any value	• The last picture has been decoded (there is no more stream to decode) but there is some frame which should be displayed. • When frame skip commend is issued. • When VPU find an error during picture header decoding.
Any value	-1	End of sequence (there is no more decoded frame and display frame)
-1	Any value	When all frame buffers are occupied by decoded picture which means that VPU cannot write the decoded frame to frame buffer because there is no more frame buffer. (This is a kind of error situation)

If there is no more stream to feed to bitstream buffer, host application should set `StreamEnd` flag in BIT_BIT_STREAM_PARAM register in order to let VPU know there is no more stream.

When empty buffer occurs, VPU asserts an interrupt signal in order to request host processor to feed more stream in bistream buffer. However, if there is no more stream to feed, it might be a dead-lock case. So host application should set `StreamEnd` flag to avoid this bad situation. When VPU meets the empty buffer and `StreamEnd` flag is set to 1, VPU returns the -2 to `indexFrameDecoded`.

3.2.4.5. Management of Displayed Buffers

VPU has some flags indicating whether frame buffer is displayed or not. The flags are set after VPU returns a display frame index automatically and VPU never uses the buffer which display flag is set. Before starting decoding process, VPU checks if there is available frame buffer and returns immediately if there is not frame buffer to be written with decoded image with current decoded index -1. (`indexFrameDecoded` == -1)

Host application can clear the flag after completion of displaying frame buffers by calling the `VPU_DecClrDispFlag()` function only when VPU is in idle state (after picture decoding has been done).

VPU_DecClrDispFlag() must use the value of indexFrameDisplay. If DecOpenParam::wtlEnable is true and VPU_DecClrDispFlag() uses the value of indexFrameDisplayForTiled, host processor might have a wrong decode result.

3.2.5. Terminating a Decoder Instance

- **Handling stream end and the last picture in bitstream buffer**

If host application meets the end of stream and sends all stream data into the bitstream buffer, it has to set `StreamEnd` flag in BS_OPTION register. This flag prevents VPU from being stalled due to stream buffer emptyness. Instead, host application can give 0 to the second argument of VPU_DecUpdateBitstreamBuffer() to inform VPU of stream end.

After sending out the last byte of stream to the bitstream buffer, host application just needs to set `StreamEnd` flag and keep calling the `VPU_DecStartOneFrame()` function. When the last output picture comes out, the decoded picture index `indexFrameDecoded` turns -1. When host application receives this index, they can easily detect the end of sequence processing.

Even though `indexFrameDecoded` is -1, there might be frames which are not displayed yet due to reordering. Host application needs to find `indexFrameDisplay` even after actual decoding operation is done. Host application still needs to call the `VPU_DecStartOneFrame()` function until the delayed output frames are completely flushed out. If there is no more output, VPU returns `indexFrameDisplay` of -1.

- **Closing current instance**

  To terminate a decoder instance, host application should releases the handle of instance and let VPU know that this instance is terminated. Host application can give the DESTROY_INST command to VPU. Instance termination can be done simply by calling the `VPU_DecClose()` function.

3.3. Other VPU Controls

3.3.1. Multiple Instances

In the multi-channel codec application, an instance is a handler to identify each task running a specific decoder or encoder. It seems that a number of tasks are running at the same time, but in fact VPU allows these multiple tasks to take turns running in a time-sharing manner.

When host processor controls VPU by using VPU API layer, the VPU API layer can handle the control of instance transition. Host processor does not need to take care of anything regarding multiple instances.

However, in case that host processor controls VPU directly by using host interface registers, some global registers should be managed by host processor when instances are switched. Before host processor gives a new command, host processor needs to backup the content of host interface registers for previously executed instance and restore the content of host interface registers for an instance to resume.

3.3.1.1. Example of Running Instances

Figure 3.3, "Example Flow of Operating Two Instances" shows when two instances are running how VPU serves them from one to another.

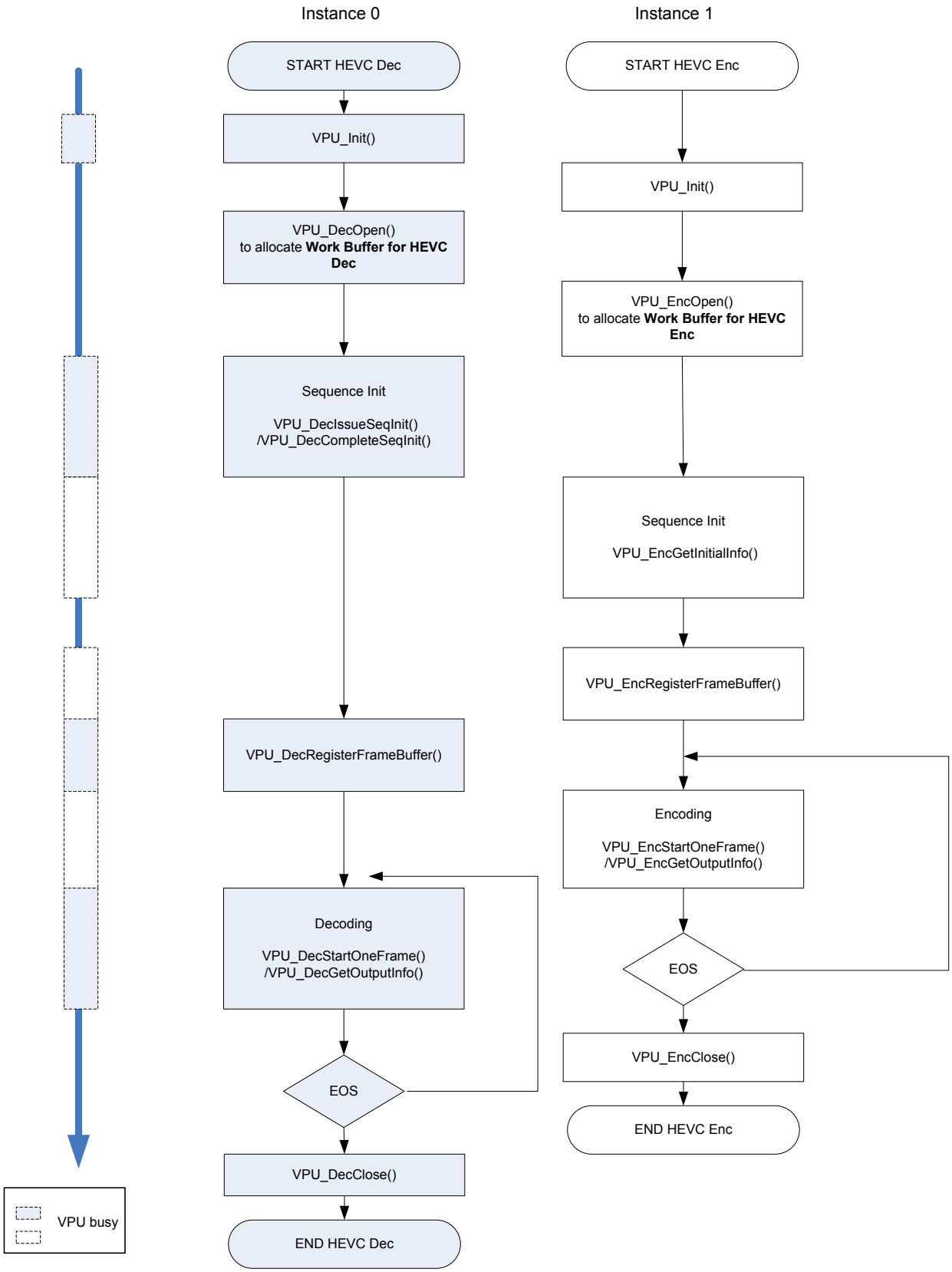


Figure 3.3. Example Flow of Operating Two Instances

As an example, VPU basically can work with instances as the following sequence. Here let us say that instance 0 is HevcDec and instance 1 is HevcEnc.

1. In Instance 0, VPU_Init() downloads firmware from host processor to the internal program memory of VPU.
2. When VPU_Init() is issued, it is not executed for Instance 1, because VPU has already been initiated in Instance 0.
3. VPU_DecOpen() allocates a work buffer for Instance 0. The information of Instance 0 are saved in work buffer.
4. In the same manner with instance 0, VPU_EncOpen() allocates a work buffer for Instance 1.
5. Sequence Init is performed for Instance 0, which returns the information of image size and DPB number.
6. Sequence Init is performed for Instance 1, which returns the information of image size and DPB number.
7. VPU_DecRegisterFrameBuffer() registers frame buffers for Instance 0 with information of the frame buffer and mvCol buffer that host processor allocates.
8. VPU_EncRegisterFrameBuffer() registers frame buffers for Instance 1 with information of the frame buffer and mvCol buffer that host processor allocates.
9. Decoding a frame starts for Instance 0.
10. Encoding a frame starts for Instance 1.

The arrow pointing downwards in the left of *Figure 3.3, "Example Flow of Operating Two Instances"* represents busy state of VPU. Two instances seem to take place in the same time. However, VPU in fact works for a single instance at the moment on a VPU command basis such as SEQ_INIT or PIC_RUN. So host application should make sure that they cannot call a VPU function in a VPU busy state. They can call another when previous function call is completed.

Note Chips&Media's reference software defines maximum four instances in vpuconfig.h. The size of memory needed for each instance is described in *Section 3.3.1.2, "Memory size for One Instance"*.

3.3.1.2. Memory size for One Instance

Please refer to the *Buffer Size* chapter in the Datasheet.

3.3.2. Sequence Change

Host processor should set CMD_SEQ_CHANGE_ENABLE_FLAG to get informed about sequence change from VPU. If any of the following SPS information changes, VPU reports the sequence change to host processor through RET_SEQ_CHANGE_RESULT register.

- General profile idc
- pic_width_in_luma_sample
- pic_height_in_luma_sample
- sps_max_dec_pic_buffering_minus1
- sps_max_reorder_pics
- sps_max_latency_increase_plus1

Note In the VPUAPI, DecOutputInfo::sequenceChanged saves these information. For more details, please refer to the *API Reference manual*.

Host application using VPUAPI should follow the steps of *Figure 3.4, "Flow Diagram of Sequence Change"* when they detect DecOutputInfo::sequenceChanged has a non-zero value after VPU_DecGetOutputInfo() is called.

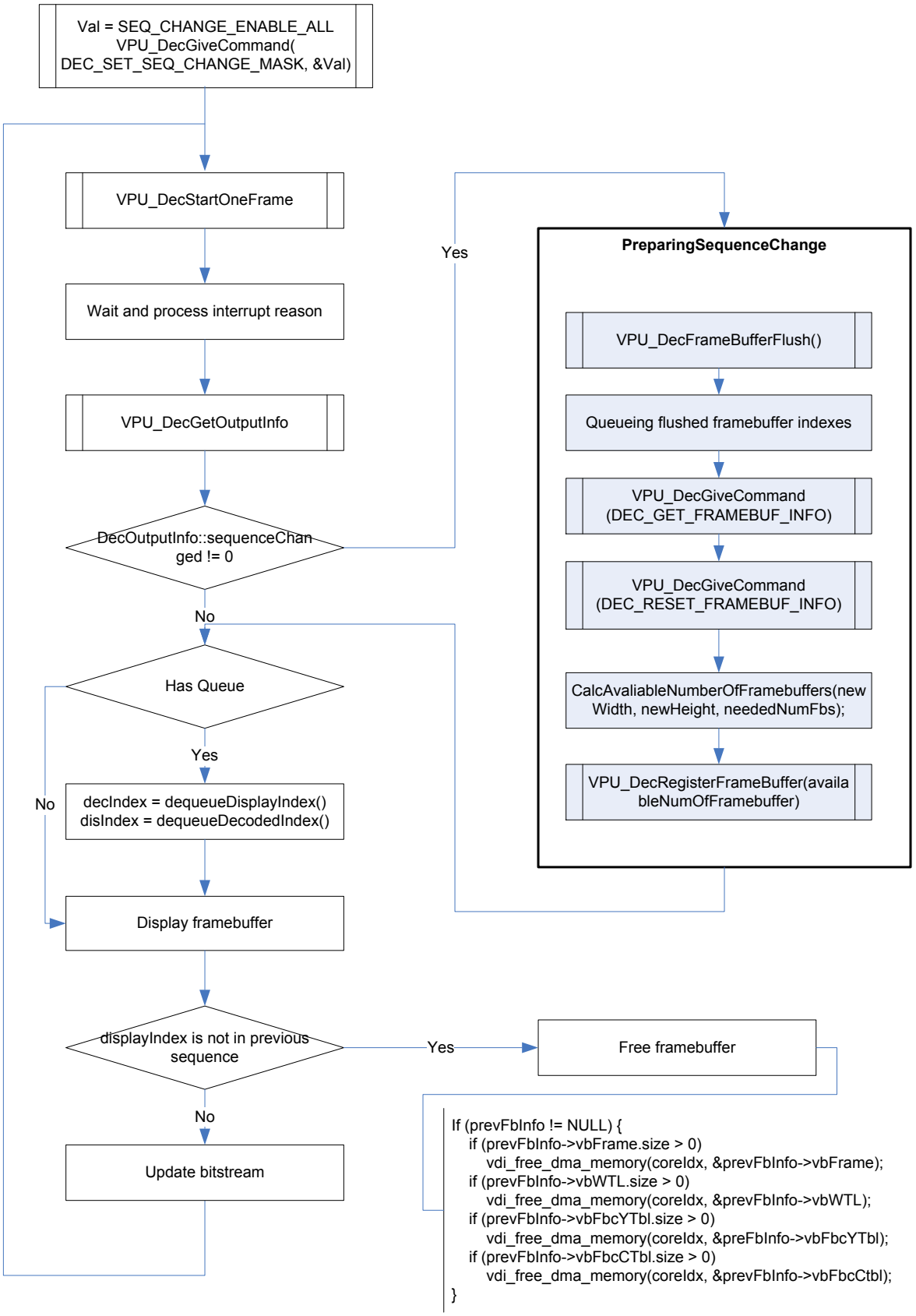


Figure 3.4. Flow Diagram of Sequence Change

Appendix A
FRAME RATE DENOMINATORS

This chapter shows the conditions generating denominators, which come from different specifications according to video coding standards. The frame rate is determined in common by following simple formular.

```
frame_rate = FrameRateNr/FrameRateDr;
```

A.1. H.265/HEVC

```
if (vui_parameters_present_flag & timing_info_present_flag)
FrameRateDr = num_units_in_tick*2;
else
FrameRateDr = -1;
```

Appendix B
FRAME RATE NUMERATORS

This chapter shows the conditions generating numerators and frame rates, which come from specification of video coding standards.

B.1. H.265/HEVC

```
if (vui_parameters_present_flag & timing_info_present_flag)
FrameRateNr = time_scale;
else
FrameRateNr = -1;
```

Appendix C
ERROR DEFINITION

C.1. System Error

The following indicates system errors that might occur while execution of command. 0x10C reports

Table C.1. System Errors

Bit Pos	Bit Position	Error Reason	Description
0	0x0000_0001	ERR_QUEUEING_FAIL	The current command failed to be queued from fullness of command queue.
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	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	The decode result cannot be retrieved, since picture decode operation has not been completed yet.
12	0x0000_1000	ERR_FLUSH_FAIL	The FLUSH_INST command failed due to incompleteness of buffer flush.
13	0x0000_2000	ERR_UNKNOWN_CMD	Unknown command error
14	0x0000_4000	ERR_UNKNOWN_CODEC_STD	Unknown codec standard error
15	0x0000_8000	ERR_UNKNOWN_QUERY_OPTION	The QUERY command failed with unknown query option.
16	0x0001_0000	Reserved	Reserved
17	0x0002_0000	Reserved	Reserved
18	0x0004_0000	Reserved	Reserved

C.2. Decode Error Information

When host processor issues the QUERY(GET_RESULT) command, VPU reports an error that has occurred during execution of DEC_PIC command. There are two severity levels that classify the VPU error.

- ERROR: VPU cannot proceed DEC_PIC command with the error and stops decoding.

- WARNING : VPU can decode it, but there is a possibility of problem occurring in the middle of decoding or taking more time in decoding.

VPU reports the error information on RET_DEC_ERR_INFO(0x1E8) register

- Decode Errors
  - SPS Syntax : There was an error in the SPS syntax.
  - PPS syntax : There was an error in the PPS syntax.
  - SLICE header syntax : There was an error in the SLICE header syntax.
  - Spec Over : Part of the syntax was out of product specifications.
  - No SPS nal : There was SPS nal in the bitstream buffer.

Note Exceptionally 0x8800 on RET_DEC_ERR_INFO[31:0] indicates that there is no response from VCE within certain time.

Table C.2. SPS Syntax Error on RET_DEC_ERR_INFO[4:0]

Error	Value	Description
SPSERR_SEQ_PARAMETER_SET_ID	1	seq_parameter_set_id golomb decode error
SPSERR_CHROMA_FORMAT_IDC	2	chroma_format_idc golomb decode error
SPSERR_PIC_WIDTH_IN_LUMA_SAMPLES	3	pic_width_in_luma_samples golomb decode error
SPSERR_PIC_HEIGHT_IN_LUMA_SAMPLES	4	pic_height_in_luma_samples golomb decode error
SPSERR_CONF_WIN_LEFT_OFFSET	5	conf_win_left_offset golomb decode error
SPSERR_CONF_WIN_RIGHT_OFFSET	6	conf_win_right_offset golomb decode error
SPSERR_CONF_WIN_TOP_OFFSET	7	conf_win_top_offset golomb decode error
SPSERR_CONF_WIN_BOTTOM_OFFSET	8	conf_win_top_offset golomb decode error
SPSERR_BIT_DEPTH_LUMA_MINUS8	9	bit_depth_luma_minus8 golomb decode error
SPSERR_BIT_DEPTH_CHROMA_MINUS8	10	bit_depth_chroma_minus8 golomb decode error
SPSERR_LOG2_MAX_PIC_ORDER_CNT_LSB_MINUS4	11	log2_max_pic_order_cnt_lsb_minus4 golomb decode error
SPSERR_SPS_MAX_DEC_PIC_BUFFERING	12	sps_max_dec_pic_buffering[i] golomb decode error
SPSERR_SPS_MAX_NUM_REORDER_PICS	13	sps_max_num_reorder_pics[i] golomb decode error
SPSERR_SPS_MAX_LATENCY_INCREASE	14	sps_sps_max_latency_increase[i] golomb decode error
SPSERR_LOG2_MIN_LUMA_CODING_BLOCK_SIZE_MINUS3	15	log2_min_luma_coding_block_size_minus3 golomb decode error
SPSERR_LOG2_DIFF_MAX_MIN_LUMA_CODING_BLOCK_SIZE	16	log2_diff_max_min_luma_coding_block_size golomb decode error
SPSERR_LOG2_MIN_TRANSFORM_BLOCK_SIZE_MINUS2	17	log2_min_transform_block_size_minus2 golomb decode error
SPSERR_LOG2_DIFF_MAX_MIN_TRANSFORM_BLOCK_SIZE	18	log2_diff_max_min_transform_block_size golomb decode error
SPSERR_MAX_TRANSFORM_HIERARCHY_DEPTH_INTER	19	max_transform_hierarchy_depth_inter golomb decode error
SPSERR_MAX_TRANSFORM_HIERARCHY_DEPTH_INTRA	20	max_transform_hierarchy_depth_intra golomb decode error
SPSERR_SCALING_LIST	21	scaling list parsing error
SPSERR_LOG2_DIFF_MIN_PCM_LUMA_CODING_BLOCK_SIZE_MINUS3	22	log2_diff_min_pcm_luma_coding_block_size_minus3 golomb decode error
SPSERR_LOG2_DIFF_MAX_MIN_PCM_LUMA_CODING_BLOCK_SIZE	23	log2_diff_max_min_pcm_luma_coding_block_size golomb decode error

Error	Value	Description
SPSERR_NUM_SHORT_TERM_REF_PIC_SETS	24	num_short_term_ref_pic_sets golomb decode error
SPSERR_NUM_LONG_TERM_REF_PICS_SPS	25	num_long_term_ref_pics_sps golomb decode error
SPSERR_RANGE_EXTENSION_FLAG	27	lack of stream forces decode error

Table C.3. PPS syntax error on RET_DEC_ERR_INFO[9:5]

Error	Value	Description
PPSERR_NUM_TILE_COLUMNS_IS_OUT_OF_ LEVEL_LIMIT	0x01	num_tile_columns went beyond the bounds of the level limit
PPSERR_NUM_TILE_ROWS_IS_OUT_OF_ LEVEL_LIMIT	0x02	num_tile_rows went beyond the bounds of the level limit

Table C.4. SLICE header syntax error on RET_DEC_ERR_INFO[15:10]

Error	Value	Description
SHERR_SLICE_PIC_PARAMETER_SET_ID	1	slice_pic_parameter_set_id decode error
SHERR_ACTIVATE_PPS	2	activate_pps decode error
SHERR_ACTIVATE_SPS	3	activate_sps decode error
SHERR_SLICE_TYPE	4	slice_type decode error
SHERR_FIRST_SLICE_IS_DEPENDENT_SLICE	5	first_slice must be independent slice
SHERR_SHORT_TERM_REF_PIC_SET_SPS_FLAG	6	short_term_ref_pic_set_sps_flag shall be equal to the first slice
SHERR_SHORT_TERM_REF_PIC_SET	7	short_term_ref_pic_set decode error
SHERR_SHORT_TERM_REF_PIC_SET_IDX	8	short_term_ref_pic_set_idx shall be equal to the first slice
SHERR_NUM_LONG_TERM_SPS	9	num_long_term_sps decode error
SHERR_NUM_LONG_TERM_PICS	10	num_long_term_pics decode error
SHERR_LT_IDX_SPS_IS_OUT_OF_RANGE	11	lt_idx_sps is out of range
SHERR_DELTA_POC_MSB_CYCLE_LT	12	delta_poc_msb_cycle_lt decode error
SHERR_NUM_REF_IDX_L0_ACTIVE_MINUS1	13	num_ref_idx_l0_active_minus1 decode error
SHERR_NUM_REF_IDX_L1_ACTIVE_MINUS1	14	num_ref_idx_l1_active_minus1 decode error
SHERR_COLLOCATED_REF_IDX	15	collocated_ref_idx decode error
SHERR_PRED_WEIGHT_TABLE	16	pred_weight_table decode error
SHERR_FIVE_MINUS_MAX_NUM_MERGE_CAND	17	five_minus_max_num_merge_cand decode error
SHERR_SLICE_QP_DELTA	18	slice_qp_delta decode error
SHERR_SLICE_QP_DELTA_IS_OUT_OF_RANGE	19	slice_qp_delta is out of range
SHERR_SLICE_CB_QP_OFFSET	20	slice_cb_qp_offset decode error
SHERR_SLICE_CR_QP_OFFSET	21	slice_cr_qp_offset decode error
SHERR_SLICE_BETA_OFFSET_DIV2	22	slice_beta_offset_div2 decode error
SHERR_SLICE_TC_OFFSET_DIV2	23	slice_tc_offset_div2 decode error
SHERR_NUM_ENTRY_POINT_OFFSETS	24	num_entry_point_offsets decode error
SHERR_OFFSET_LEN_MINUS1	25	offset_len_minus1 decode error
SHERR_SLICE_SEGMENT_HEADER_ EXTENSION_LENGTH	26	slice_segment_header_extension_length decode error
SHERR_DPB_OVERFLOW	28	dpb overflow

Table C.5. SPEC over error on RET_DEC_ERR_INFO[23:16]

Error	Value	Description
SPEC_OVER_PICTURE_WIDTH_SIZE	0x01	pic_width_in_luma_samples or pic_height_in_luma_samples
SPEC_OVER_PICTURE_HEIGHT_SIZE	0x02	pic_width_in_luma_samples or pic_height_in_luma_samples
SPEC_OVER_CHROMA_FORMAT	0x04	chroma_format_idc
SPEC_OVER_BIT_DEPTH	0x08	bit_depth_luma_minus8 or bit_depth_chroma_minus

Table C.6. ETC error on RET_DEC_ERR_INFO[31:23]

Error	Value	Description
ETC_INIT_SEQ_SPS_NOT_FOUND	0x01	SPS not found
ETC_DEC_PIC_VCL_NOT_FOUND	0x02	VCL not found
ETC_NO_VALID_SLICE_IN_AU	0x08	First VCL of the next AU was detected while parsing the slice header.

C.3. Decode Warning Information

VPU reports the warning information on RET_DEC_WARN_INFO(0x1E4) regsiter.

- Decode Warnings
  - SPS Syntax : There was an warning in the SPS syntax.
  - PPS syntax : There was an warning in the PPS syntax.
  - SLICE header syntax : There was an warning in the SLICE header syntax.
  - ETC : Other than the three above cases such as missing slice/PS(Parameter Set) to activate.
  - SPEC over : Part of the syntax was out of product specifications.

Table C.7. SPS syntax warning on RET_DEC_WARN_INFO[5:0]

Error	Value	Description
SPSERR_MAX_SUB_LAYERS_MINUS1	0x01	sps_max_sub_layer_minus1 shall be 0 to 6.
SPSERR_GENERAL_RESERVED_ZERO_44BITS	0x02	general_reserved_zero_44bits shall be 0.
SPSERR_RESERVED_ZERO_2BITS	0x02	reserved_zero_2bits shall be 0.
SPSERR_SUB_LAYER_RESERVED_ZERO_44BITS	0x02	sub_layer_reserved_zero_44bits shall be 0.
SPSERR_GENERAL_LEVEL_IDC	0x04	general_level_idc shall have one of level of Table A.1
SPSERR_SPS_MAX_DEC_PIC_BUFFERING_ VALUE_OVER	0x08	sps_max_dec_pic_buffering[i] <= MaxDpbSize
SPSERR_RBSP_TRAILING_BITS	0x10	trailing bits shall be 1000… pattern, 7.3.2.1
SPSERR_CURRENTLY_DECODED_SPS_ERROR	0x20	This warning is reported when the currently decoded SPS has an error, but the same sps_id of SPS that was previously decoded without error already exists. If the currently decoded SPS has an error and there is no same sps_id of SPS that was previously decoded, VPU reports an error(not warning) on RET_DEC_ERR_INFO register.

Table C.8. PPS syntax warning on RET_DEC_WARN_INFO[9:6]

Error	Value	Description
PPSERR_RBSP_TRAILING_BITS	0x04	trailing bits shall be 1000… pattern, 7.3.2.11

Error	Value	Description
PPSERR_CURRENTLY_DECODED_SPS_ERROR	0x08	current decoded pps has error

Table C.9. SLICE header syntax warning on RET_DEC_WARN_INFO[12:10]

Error	Value	Description
SHERR_FIRST_SLICE_SEGMENT_IN_PIC_FLAG	0x01	first_slice_segment_in_pic_flag shall be 1 at first slice
SHERR_NO_OUTPUT_OF_PRIOR_PICS_FLAG	0x02	no_output_of_prior_pics_flag has different value at same AU
SHERR_PIC_OUTPUT_FLAG	0x04	pic_output_flag has different value at same AU)

Table C.10. ETC warning on RET_DEC_WARN_INFO[14:13]

Error	Value	Description
ETC_INIT_SEQ_VCL_NOT_FOUND	0x01	VCL not found
ETC_MISSING_REFERENCE_PICTURE	0x02	Reference picture was missing
ETC_WRONG_TEMPORAL_ID	0x04	The picture has a wrong temporal ID.
ETC_ERROR_PICTURE_IS_REFERENCED	0x10	The picture was decoded with error reference picture.

Table C.11. SPEC over warning on RET_DEC_WARN_INFO[16:15]

Error	Value	Description
SPEC_OVER_PROFILE	0x1	general_profile_idc and general_profile_compatibility_flag is out of specification. In the product whose product code is 4100 on the register 0x1044, general_profile_idc and general_profile_compatibility_flag should specify Main profile. In the product whose product code is 4102 on the register 0x1044, general_profile_idc and general_profile_compatibility_flag should specify Main or Main10 profile.
SPEC_OVER_LEVEL	0x2	general_level_idc is out of specification.

Appendix D
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.

D.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 V-CPU pmem/dmem are removed. Then V-CPU 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.

D.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.

D.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.

D.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.

D.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 D.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 E
VP9 SUPERFRAME

E.1. Superframe Syntax

Superframe is a way that vp9 uses to consolidate multiple compressed video frames into one single chunk. The superframe index is stored in the last up to 34 bytes of a chunk. To determine whether a chunk of data has a superframe index, check the last byte in the chunk for the marker 3 bits (0b110xxxxx):

If the marker bits are set, apply a mask to the byte

- mask 2 bits (0bxxx11xxx) and add 1 to get the size in bytes for each frame size (1 to 4)
- mask 3 bits (0bxxxxx111) add add 1 to get the # of frame sizes encoded in the index (1 to 8)

```
Syntax                                                  Descriptor
_____________________________________________________________________
superframe_index() {
    SUPERFRAME_MARKER /* equal to 0b110 */              f(3)
    frame_size_length_minus_one                         u(2)
    num_frames_minus_one                                u(3)
    for (i = 0? i <= num_frames_minus_one? ++i)
        frame_sizes[i]                                  u_bytes(v)
    SUPERFRAME_MARKER /* equal to 0b110 */              f(3)
    frame_size_length_minus_one                         u(2)
    num_frames_minus_one                                u(3)
}
```

Syntax elements

- `frame_size_length_minus_one` plus one specifies the size in bytes for each frame.
- `num_frames_minus_one` plus one specifies number of frames contained in this superframe.
- `frame_sizes[i]` specifies the size of ith frame. Every frame_sizes[i] has frame_size_length_minus_one + 1 bytes.

Calculate the total size of the index as follows:

```
2 + number of frame sizes encoded * number of bytes to hold each frame size
```

2 represents the one byte holding a marker byte that is duplicated at the start of the index and at the end of the index.

Figure E.1, “superframe_index()” shows representation of a superframe marked chunk with marker 3 bits(110), frame_size_length_minus_one(00), and num_frames_minus_one frame size(010)

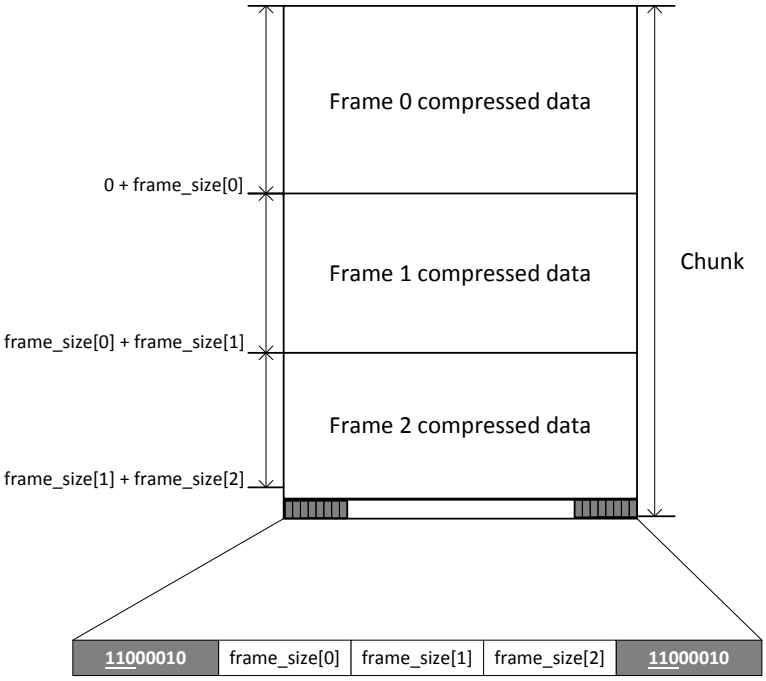


Figure E.1. superframe_index()

E.2. Parsing Frame Size from Superframe

For decoding VP9 stream, VPU can read only a frame unit of compressed data from the bitstream buffer, which is called pic-end mode as one of the bitream buffer operation modes. Therefore, host needs to feed a single frame data before issuing PIC_RUN command. If there is a chunk with a superframe marker, host should handle this - reading each frame size from superframe syntax and writing a single frame to the bitstream buffer in frame-by-frame manner.

This is the relevant part of the code we provide for example.

```
// has super frame
if ((last_byte & 0xE0) == 0xC0)
{
    int frame_cnt = (last_byte & 0x7) + 1;
    int len       = ((last_byte >> 3) & 0x3) + 1;

    uint8_t* ptr       = bs_wr_ptr - 1 - len*frame_cnt;

    for (i = 0, k = 0 ; i < frame_cnt ; i++)
    {
        frame_size[i] = 0;

        for (j = 0 ; j < len ; j++, k++)
            frame_size[i] = (frame_size[i] << 8) | ptr[k];
    }
}
else
{
   frame_cnt = 1;
   frame_size[0] = bs_wr_ptr - bs_rd_ptr;
}
```

Note For the detail, please refer to the VP9ParseSuperframe() function in sample/helper/bistream/bsfeeder_framesize_impl.c

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 products from low-end 1080p HD mobile solutions to high-end 4K 60fps dual-core solution for e.g. Ultra-HD TV or even brandnew HEVC/H.265 and VP9 decoder IP to fulfill target demands from various multimedia device markets.

Chips&Media's IPs are proven in over a hundred million of 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, Taiwan, and the USA. To find further information, please visit the company's web site at *http://www.chipsnmedia.com*