

Intel® Iris® Xe MAX Graphics Open Source

Programmer's Reference Manual

For the 2020 Discrete GPU formerly named "DG1"

Volume 2a: Command Reference: Instructions

February 2021, Revision 1.0



Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks.

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

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

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

3DPRIMITIVE	1
3DSTATE_3D_MODE.....	7
3DSTATE_AA_LINE_PARAMETERS.....	10
3DSTATE_AMFS.....	12
3DSTATE_AMFS_TEXTURE_POINTERS	13
3DSTATE_BINDING_TABLE_POINTERS_DS.....	14
3DSTATE_BINDING_TABLE_POINTERS_GS.....	15
3DSTATE_BINDING_TABLE_POINTERS_HS.....	16
3DSTATE_BINDING_TABLE_POINTERS_PS	17
3DSTATE_BINDING_TABLE_POINTERS_VS	18
3DSTATE_BINDING_TABLE_POOL_ALLOC	19
3DSTATE_BLEND_STATE_POINTERS	21
3DSTATE_CC_STATE_POINTERS.....	22
3DSTATE_CHROMA_KEY	23
3DSTATE_CLEAR_PARAMS.....	25
3DSTATE_CLIP	26
3DSTATE_CONSTANT_ALL	27
3DSTATE_CONSTANT_DS.....	29
3DSTATE_CONSTANT_GS.....	31
3DSTATE_CONSTANT_HS.....	33
3DSTATE_CONSTANT_PS	35
3DSTATE_CONSTANT_TS_POINTER.....	37
3DSTATE_CONSTANT_VS	38
3DSTATE_CPS_POINTERS	40
3DSTATE_DEPTH_BOUNDS.....	41
3DSTATE_DEPTH_BUFFER	42
3DSTATE_DRAWING_RECTANGLE	51
3DSTATE_DS	54
3DSTATE_GS	55
3DSTATE_HIER_DEPTH_BUFFER	56
3DSTATE_HS	57
3DSTATE_INDEX_BUFFER.....	58



3DSTATE_LINE_STIPPLE	59
3DSTATE_MULTISAMPLE	61
3DSTATE_POLY_STIPPLE_OFFSET	62
3DSTATE_POLY_STIPPLE_PATTERN	64
3DSTATE_PRIMITIVE_REPLICATION	65
3DSTATE_PS_BLEND	66
3DSTATE_PS_EXTRA	67
3DSTATE_PS	68
3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS	69
3DSTATE_PTBR_MARKER	71
3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS	72
3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS	74
3DSTATE_PTBR_TILE_PASS_INFO	76
3DSTATE_PTBR_TILE_SELECT	80
3DSTATE_PUSH_CONSTANT_ALLOC_DS	81
3DSTATE_PUSH_CONSTANT_ALLOC_GS	83
3DSTATE_PUSH_CONSTANT_ALLOC_HS	85
3DSTATE_PUSH_CONSTANT_ALLOC_PS	87
3DSTATE_PUSH_CONSTANT_ALLOC_VS	89
3DSTATE_RASTER	91
3DSTATE_SAMPLE_MASK	94
3DSTATE_SAMPLE_PATTERN	95
3DSTATE_SAMPLER_STATE_POINTERS_DS	105
3DSTATE_SAMPLER_STATE_POINTERS_GS	106
3DSTATE_SAMPLER_STATE_POINTERS_HS	107
3DSTATE_SAMPLER_STATE_POINTERS_PS	108
3DSTATE_SAMPLER_STATE_POINTERS_VS	109
3DSTATE_SBE	110
3DSTATE_SBE_SWIZ	111
3DSTATE_SCISSOR_STATE_POINTERS	112
3DSTATE_SF	113
3DSTATE_SLICE_TABLE_STATE_POINTERS	115
3DSTATE_SO_BUFFER_INDEX_0	116
3DSTATE_SO_BUFFER_INDEX_1	117

3DSTATE_SO_BUFFER_INDEX_2	118
3DSTATE_SO_BUFFER_INDEX_3	119
3DSTATE_SO_BUFFER	120
3DSTATE_SO_DECL_LIST	123
3DSTATE_STENCIL_BUFFER	126
3DSTATE_STREAMOUT	127
3DSTATE_SUBSLICE_HASH_TABLE	128
3DSTATE_TE	130
3DSTATE_URB_ALLOC_DS	131
3DSTATE_URB_ALLOC_GS	133
3DSTATE_URB_ALLOC_HS	135
3DSTATE_URB_ALLOC_VS	137
3DSTATE_URB_DS	139
3DSTATE_URB_GS	140
3DSTATE_URB_HS	141
3DSTATE_URB_VS	142
3DSTATE_VERTEX_BUFFERS	144
3DSTATE_VERTEX_ELEMENTS	146
3DSTATE_VF_COMPONENT_PACKING	148
3DSTATE_VF_INSTANCING	150
3DSTATE_VF	151
3DSTATE_VF_SGVS_2	154
3DSTATE_VF_SGVS	156
3DSTATE_VF_STATISTICS	158
3DSTATE_VF_TOPOLOGY	159
3DSTATE_VIEWPORT_STATE_POINTERS_CC	160
3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP	161
3DSTATE_VS	162
3DSTATE_WM_CHROMAKEY	163
3DSTATE_WM_DEPTH_STENCIL	164
3DSTATE_WM_HZ_OP	166
3DSTATE_WM	168
A64 Byte Scattered Read MSD	169
A64 Byte Scattered Write MSD	171



A64 Dword Scattered Read MSD	173
A64 Dword Scattered Write MSD	175
A64 Dword Untyped Atomic Float with Return Data Operation MSD	177
A64 Dword Untyped Atomic Float Write Only Operation MSD	179
A64 Dword Untyped Atomic Integer with Return Data Operation MSD	181
A64 Dword Untyped Atomic Integer Write Only Operation MSD	183
A64 Hword Block Read MSD	185
A64 Hword Block Write MSD	187
A64 Oword Aligned Block Read MSD.....	189
A64 Oword Aligned Block Write MSD.....	190
A64 Oword Block Read MSD	191
A64 Oword Block Write MSD	193
A64 Qword Scattered Read MSD	195
A64 Qword Scattered Write MSD	197
A64 Qword Untyped Atomic Integer with Return Data Operation MSD	199
A64 Qword Untyped Atomic Integer Write Only Operation MSD	201
A64 Untyped Surface Read MSD.....	203
A64 Untyped Surface Write MSD.....	204
A64 Word Untyped Atomic Float with Return Data Operation MSD	205
A64 Word Untyped Atomic Float Write Only Operation MSD	207
A64 Word Untyped Atomic Integer with Return Data Operation MSD	209
A64 Word Untyped Atomic Integer Write Only Operation MSD	211
Addition	213
Addition with Carry	218
Arithmetic Shift Right	223
Average	228
AVP_BSD_OBJECT.....	233
AVP_IND_OBJ_BASE_ADDR_STATE	235
AVP_INLOOP_FILTER_STATE	237
AVP_INTER_PRED_STATE	250
AVP_PIC_STATE	257
AVP_PIPE_BUF_ADDR_STATE.....	288
AVP_PIPE_MODE_SELECT	299
AVP_SEGMENT_STATE.....	302

AVP_SURFACE_STATE	307
AVP_TILE_CODING	313
AVP_VD_CONTROL_STATE	323
Barrier	324
Bit Field Extract	326
Bit Field Insert 1	331
Bit Field Insert 2	336
Bit Field Reverse	341
Branch Converging	345
Branch Diverging	349
Break	353
Byte Scattered Read MSD	357
Byte Scattered Write MSD	359
Call	361
Call Absolute	365
Compare	369
Compare NaN	374
Conditional Select	379
Constant Cache Dword Scattered Read MSD	384
Constant Cache Oword Aligned Block Read MSD	386
Constant Cache Oword Block Read MSD	387
Continue	388
Count Bits Set	392
Dot Product 4 Accumulate	396
Dword Atomic Counter with Return Data Operation MSD	401
Dword Atomic Counter Write Only Operation MSD	403
Dword Scattered Read MSD	405
Dword Scattered Write MSD	407
Dword Typed Atomic Integer with Return Data Operation MSD	409
Dword Typed Atomic Integer Write Only Operation MSD	411
Dword Untyped Atomic Float with Return Data Operation MSD	413
Dword Untyped Atomic Float Write Only Operation MSD	415
Dword Untyped Atomic Integer with Return Data Operation MSD	417
Dword Untyped Atomic Integer Write Only Operation MSD	419



Else	421
End If	425
Extended Math Function	429
Find First Bit from LSB Side	435
Find First Bit from MSB Side.....	439
Fraction	444
Goto	448
GPGPU_CSR_BASE_ADDRESS	452
GPGPU_WALKER	454
Half Precision HI8DS Render Target Write MSD.....	458
Half Precision LO8DS Render Target Write MSD	460
Half Precision REP16 Render Target Write MSD.....	463
Half Precision SIMD8 Render Target Write MSD.....	466
Half Precision SIMD16 Render Target Write MSD.....	469
Halt	472
HCP_BSD_OBJECT.....	476
HCP_FQM_STATE.....	478
HCP_IND_OBJ_BASE_ADDR_STATE	481
HCP_PAK_INSERT_OBJECT	485
HCP_PAK_OBJECT	490
HCP_PALETTE_INITIALIZER_STATE	493
HCP_PIC_STATE	495
HCP_PIPE_BUF_ADDR_STATE.....	527
HCP_PIPE_MODE_SELECT	534
HCP_QM_STATE.....	540
HCP_REF_IDX_STATE.....	543
HCP_SFC_LOCK	545
HCP_SFC_STATE.....	547
HCP_SLICE_STATE.....	548
HCP_SURFACE_STATE.....	563
HCP_TILE_CODING	568
HCP_TILE_STATE.....	574
HCP_VP9_PAK_OBJECT	576
HCP_VP9_PIC_STATE.....	579

HCP_VP9_SEGMENT_STATE	600
HCP_WEIGHTOFFSET_STATE.....	604
HEVC_SFC_AVS_CHROMA_Coeff_Table	606
HEVC_SFC_AVS_LUMA_Coeff_Table	607
HEVC_SFC_AVS_STATE.....	608
HEVC_SFC_FRAME_START	609
HEVC_SFC_IEF_STATE.....	610
HEVC_VP9_RDOQ_STATE.....	611
HI8DS Render Target Write MSD.....	613
If	615
Illegal	619
Join	620
Jump Indexed	624
L3_CONTROL.....	628
Leading Zero Detection.....	631
LO8DS Render Target Write MSD.....	635
Logic And	637
Logic Not.....	642
Logic Or	646
Logic Xor	651
MEDIA_CURBE_LOAD.....	656
MEDIA_INTERFACE_DESCRIPTOR_LOAD	658
MEDIA_OBJECT_GRPID	660
MEDIA_OBJECT.....	663
MEDIA_OBJECT_WALKER	666
MEDIA_STATE_FLUSH	673
MEDIA_VFE_STATE	675
Media Block Read MSD	679
Media Block Write MSD	680
Memory Fence MSD	681
MFC_AVC_PAK_OBJECT	683
MFC_JPEG_HUFF_TABLE_STATE.....	685
MFC_JPEG_SCAN_OBJECT	687
MFC_MPEG2_PAK_OBJECT	690



MFC_MPEG2_SLICEGROUP_STATE	692
MFD_AVC_BSD_OBJECT	699
MFD_AVC_DPB_STATE.....	701
MFD_AVC_PICID_STATE.....	704
MFD_AVC_SLICEADDR.....	706
MFD_IT_OBJECT.....	709
MFD_JPEG_BSD_OBJECT.....	713
MFD_MPEG2_BSD_OBJECT.....	715
MFD_VC1_BSD_OBJECT	717
MFD_VC1_LONG_PIC_STATE	720
MFD_VC1_SHORT_PIC_STATE	733
MFD_VP8_BSD_OBJECT	741
MFV_AVC_DIRECTMODE_STATE.....	747
MFV_AVC_IMG_STATE	749
MFV_AVC_REF_IDX_STATE	768
MFV_AVC_SLICE_STATE	771
MFV_AVC_WEIGHTOFFSET_STATE	784
MFV_BSP_BUF_BASE_ADDR_STATE.....	786
MFV_DBK_OBJECT	792
MFV_FQM_STATE	798
MFV_IND_OBJ_BASE_ADDR_STATE.....	800
MFV_JPEG_HUFF_TABLE_STATE.....	804
MFV_JPEG_PIC_STATE.....	806
MFV_MPEG_TS_CONTROL command.....	812
MFV_MPEG2_PIC_STATE.....	814
MFV_PAK_INSERT_OBJECT.....	827
MFV_PIPE_BUF_ADDR_STATE	831
MFV_PIPE_MODE_SELECT	852
MFV_QM_STATE	857
MFV_STATE_POINTER.....	859
MFV_STITCH_OBJECT.....	861
MFV_SURFACE_STATE.....	863
MFV_VC1_DIRECTMODE_STATE	871
MFV_VC1_PRED_PIPE_STATE	873

MFX_VP8_BSP_BUF_BASE_ADDR_STATE	879
MFX_VP8_Encoder_CFG	882
MFX_VP8_PAK_OBJECT.....	893
MFX_VP8_PIC_STATE	895
MFX_WAIT	919
MI_ARB_CHECK	920
MI_ARB_ON_OFF	922
MI_ATOMIC.....	924
MI_BATCH_BUFFER_END	929
MI_BATCH_BUFFER_START	930
MI_CONDITIONAL_BATCH_BUFFER_END	937
MI_COPY_MEM_MEM	941
MI_COPY_MEM_MEM	943
MI_COPY_MEM_MEM	945
MI_COPY_MEM_MEM	947
MI_DISPLAY_FLIP	949
MI_FLUSH_DW.....	954
MI_FLUSH_DW	957
MI_FLUSH_DW	960
MI_FORCE_WAKEUP.....	963
MI_LOAD_REGISTER_IMM	966
MI_LOAD_REGISTER_MEM	970
MI_LOAD_REGISTER_REG	973
MI_LOAD_SCAN_LINES_EXCL	977
MI_LOAD_SCAN_LINES_EXCL	979
MI_LOAD_SCAN_LINES_INCL	981
MI_LOAD_SCAN_LINES_INCL	983
MI_MATH	985
MI_MATH	986
MI_MATH	987
MI_MATH	988
MI_NOOP	989
MI_NOOP	990
MI_NOOP	991



MI_NOOP	992
MI_PREDICATE	993
MI_REPORT_HEAD	995
MI_REPORT_HEAD	996
MI_REPORT_HEAD	997
MI_REPORT_HEAD	998
MI_REPORT_PERF_COUNT	999
MI_RS_STORE_DATA_IMM	1001
MI_SEMAPHORE_SIGNAL	1003
MI_SEMAPHORE_WAIT	1005
MI_SET_CONTEXT	1009
MI_SET_PREDICATE	1012
MI_STORE_DATA_IMM	1014
MI_STORE_DATA_INDEX	1017
MI_STORE_REGISTER_MEM	1019
MI_SUSPEND_FLUSH	1022
MI_SUSPEND_FLUSH	1023
MI_SUSPEND_FLUSH	1024
MI_SUSPEND_FLUSH	1025
MI_TOPOLOGY_FILTER	1026
MI_UPDATE_GTT	1027
MI_USER_INTERRUPT	1029
MI_USER_INTERRUPT	1030
MI_USER_INTERRUPT	1031
MI_USER_INTERRUPT	1032
MI_WAIT_FOR_EVENT	1033
MI_WAIT_FOR_EVENT_2	1037
Monitor Event	1040
Monitor No Event	1041
Move	1042
Move Indexed	1047
Multiply	1053
Multiply Accumulate	1058
Multiply Accumulate High	1063

Multiply Add.....	1069
No Operation.....	1074
Oword Aligned Block Read MSD	1075
Oword Block Read MSD	1077
Oword Block Write MSD	1079
PIPE_CONTROL.....	1081
PIPELINE_SELECT	1091
Read Surface Info MSD	1094
REP16 Render Target Write MSD.....	1095
Reserved Instruction0.....	1097
Reserved Instruction1.....	1098
Reserved Instruction2.....	1099
Reserved Instruction3.....	1100
Reserved Instruction4.....	1101
Reserved Instruction5.....	1102
Reserved Instruction6.....	1103
Reserved Instruction7.....	1104
Reserved Instruction8.....	1105
Reserved Instruction9.....	1106
Reserved Instruction10	1107
Reserved Instruction11	1108
Reserved Instruction12	1109
Reserved Instruction13	1110
Reserved Instruction14	1111
Reserved Instruction15	1112
Reserved Instruction16	1113
Reserved Instruction17	1114
Reserved Instruction18	1115
Reserved Instruction19	1116
Reserved Instruction20	1117
Reserved Instruction21	1118
Reserved Instruction22	1119
Reserved Instruction23	1120
Reserved Instruction24	1121



Reserved Instruction25	1122
Reserved Instruction26	1123
Return.....	1124
Rotate Left.....	1128
Rotate Right	1133
Round Down.....	1138
Round to Nearest or Even.....	1142
Round to Zero	1146
Round Up.....	1150
Sampler Cache Media Block Read MSD	1154
Sampler Cache Oword Aligned Block Read MSD.....	1155
Scratch Block Read MSD	1156
Scratch Block Write MSD	1158
Select.....	1160
Send Message.....	1165
Send Message Conditional.....	1173
SFC_AVS_CHROMA_Coeff_Table	1179
SFC_AVS_LUMA_Coeff_Table	1180
SFC_AVS_STATE.....	1181
SFC_FRAME_START	1182
SFC_IEF_STATE.....	1183
SFC_LOCK	1184
SFC_STATE.....	1186
Shift Left.....	1187
Shift Right.....	1192
Signal Event	1197
SIMD8 Render Target Read MSD.....	1198
SIMD8 Render Target Write MSD.....	1200
SIMD16 Render Target Read MSD.....	1202
SIMD16 Render Target Write MSD.....	1204
STATE_BASE_ADDRESS.....	1207
STATE_COMPUTE_MODE.....	1216
STATE_SIP	1219
Subtraction with Borrow.....	1221

Synchronize	1226
Typed Surface Read MSD.....	1231
Typed Surface Write MSD.....	1233
Untyped Surface Read MSD	1235
Untyped Surface Write MSD	1237
URB Dword Read MSD	1239
URB Dword Write MSD	1241
URB Masked Dword Write MSD.....	1243
VD_CONTROL_STATE.....	1245
VD_PIPELINE_FLUSH	1246
VDENC_CONTROL_STATE.....	1248
VDENC_DS_REF_SURFACE_STATE.....	1249
VDENC_PIPE_BUF_ADDR_STATE.....	1250
VDENC_PIPE_MODE_SELECT	1255
VDENC_REF_SURFACE_STATE	1258
VDENC_SRC_SURFACE_STATE.....	1259
VDENC_WEIGHTSOFFSETS_STATE	1261
VEBOX_STATE.....	1267
VEBOX_SURFACE_STATE	1276
VEBOX_TILING_CONVERT	1284
Wait for Event	1286
While	1287
Word Atomic Counter with Return Data Operation MSD.....	1291
Word Atomic Counter Write Only Operation MSD	1293
Word Typed Atomic Integer with Return Data Operation MSD	1295
Word Typed Atomic Integer Write Only Operation MSD	1297
Word Untyped Atomic Float with Return Data Operation MSD.....	1299
Word Untyped Atomic Float Write Only Operation MSD.....	1301
Word Untyped Atomic Integer with Return Data Operation MSD	1303
Word Untyped Atomic Integer Write Only Operation MSD.....	1305
XY_BLOCK_COPY_BLT	1307
XY_COLOR_BLT	1312
XY_FAST_COLOR_BLT	1314
XY_FAST_COPY_BLT	1318



XY_FULL_BLT	1321
XY_FULL_IMMEDIATE_PATTERN_BLT	1324
XY_FULL_MONO_PATTERN_BLT	1327
XY_FULL_MONO_PATTERN_MONO_SRC_BLT	1330
XY_FULL_MONO_SRC_BLT	1333
XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT	1336
XY_MONO_PAT_BLT.....	1339
XY_MONO_PAT_FIXED_BLT	1342
XY_MONO_SRC_COPY_BLT.....	1345
XY_MONO_SRC_COPY_IMMEDIATE_BLT	1348
XY_PAT_BLT_IMMEDIATE.....	1351
XY_PAT_BLT	1354
XY_PAT_CHROMA_BLT_IMMEDIATE.....	1356
XY_PAT_CHROMA_BLT	1359
XY_PIXEL_BLT.....	1361
XY_SCANLINES_BLT	1362
XY_SETUP_BLT	1363
XY_SETUP_CLIP_BLT	1366
XY_SETUP_MONO_PATTERN_SL_BLT	1367
XY_SRC_COPY_BLT	1370
XY_SRC_COPY_CHROMA_BLT	1373
XY_TEXT_BLT.....	1376
XY_TEXT_IMMEDIATE_BLT	1378

3DPRIMITIVE

3DPRIMITIVE			
Source:	RenderCS		
Length Bias:	2		
<p>The 3DPRIMITIVE command is used to submit 3D primitives to be processed by the 3D pipeline. Typically the processing results in rendering pixel data into the render targets, but this is not required. The parameters passed in this command are forwarded to the Vertex Fetch function. The Vertex Fetch function will use this information to generate vertex data structures and store them in the URB. These vertices are then passed down the 3D pipeline.</p>			
Programming Notes			
<p>If the threads spawned by this command are required to observe memory writes performed by threads spawned from a previous command, software must precede this command with a command that performs a (preferably pipelined) memory flush (e.g., 3D_PIPECONTROL).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	3h 3DPRIMITIVE
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	0h 3DPRIMITIVE
		Format:	OpCode
	15	Reserved	
	14	Reserved	
Format:		MBZ	
13	Reserved		
	Format:	MBZ	
12	POSH Enable POSH functionality is enabled for this 3DPRIMITIVE command when this bit is set. When executed from the POCS command stream, the 3DPRIMITIVE command is processed and visibility information is recorded. When subsequently executed from the RCS command stream, the previously recorded visibility information is used to optimize the processing for this 3DPRIMITIVE command. Setting this bit will also (temporarily -- for the 3DPRIMITIVE command) inhibit		

3DPRIMITIVE

modification of the following statistics counters when the 3DPRIMITIVE command is executed from the RCS command stream, even if these statistics counters are enabled via VF/VS state. This is done to prevent RCS from double-counting statistics that have already been counted by POCS.

- IA_VERTICES_COUNT
- IA_PRIMITIVES_COUNT
- VS_INVOCATION_COUNT

When this bit is clear, this 3DPRIMITIVE command is ignored by the POCS command stream, and when executed in the RCS command stream the command is processed without use of visibility information (as none will have been recorded by the POCS command stream) and with statistics counted as specified by VF/VS state.

Programming Notes

This bit must be clear (POSH disabled) when any one of the following conditions are met:

- A topology type other than: 3DPRIM_TRILIST, 3DPRIM_TRISTRIP, 3DPRIM_TRISTRIP_REVERSE
- UAV Coherency Required and/or Accesses UAV set in 3DSTATE_VS
- Tessellation enabled
- Geometry Shader enabled
- StreamOut enabled

For each 3DPRIMITIVE command that SW submits to the POCS command stream with POSH enabled, SW shall also submit an identical 3DPRIMITIVE command with POSH enabled to the RCS command stream, in the same sequence as submitted to the POCS command stream. In addition to the parameters passed in the 3DPRIMITIVE command fields, the following states shall also match:

- 3DSTATE_VF: States controlling cut index processing must match
- For RANDOM indexing, the Index Buffer states must match

11 Extended Parameters Present

Format:	Boolean
---------	---------

If true, three additional DWords containing XP0, XP1 and XP2 parameters are included in this command. Depending on the setting of Indirect Parameter Enable, XP0-2 parameters (sourced either from the inline XP0-2 DWords or indirectly from the 3DPRIM_XP0-2 registers) are passed to the VF stage as possible sources for VF SGV insertion.

If false, XP0-2 DWords are not included in this command and instead XP0-2 values default to zero if enabled as sources in VF SGV insertion.

10 Indirect Parameter Enable

Format:	Enable
---------	--------

3DPRIMITIVE

		<p>If set, the values in DW 2-5 are ignored and replaced by the current values of the corresponding 3DPRIM_xxx MMIO registers:</p> <ul style="list-style-type: none"> • 3DPRIM_VERTEX_COUNT (instead of DW2: VertexCountPerInstance) • 3DPRIM_START_VERTEX (instead of DW3: StartVertexLocation) • 3DPRIM_INSTANCE_COUNT (instead of DW4: InstanceCount) • 3DPRIM_START_INSTANCE (instead of DW5: StartInstanceLocation) • 3DPRIM_BASE_VERTEX (instead of DW6: BaseVertexLocation) <p>Indirect Parameter Enable and End Offset Enable shall not be ENABLED at the same time, or behavior is UNDEFINED.</p> <p>If set and Extended Parameters Present is true, the current contents of the 3DPRIM_XP0-2 MMIO registers are passed to the VF stage as possible sources for VF SGV insertion and the Extended Parameter 0-2 values included in this command are ignored.</p>																
	9	<p>UAV Coherency Required</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>SW will be required to set this bit if there is the possibility of sharing a UAV from a previous 3DPRIMITIVE command. If set, this command may cause a flush due to UAV coherency requirements. If none of the shaders have UAV access enabled, then this bit is ignored.</p>			Format:	U1												
Format:	U1																	
	8	<p>Predicate Enable</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Enable</td> </tr> </table> <p>If set, this command is executed (or not) depending on the current value of the MI Predicate internal state bit. This command is ignored only if PredicateEnable is set and the Predicate state bit is 0.</p>			Format:	Enable												
Format:	Enable																	
	7:0	<p>DWord Length</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">=n</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 10%;">Value</th> <th style="width: 10%;">Name</th> <th style="width: 50%;">Programming Notes</th> <th style="width: 30%;">Exists If</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">5h</td> <td style="text-align: center;">5</td> <td></td> <td>[Extended Parameter Enable] == 'false'</td> </tr> <tr> <td style="text-align: center;">8h</td> <td style="text-align: center;">[Default]</td> <td>When Extended Parameter Enable is true, three Dwords containing Extended Parameter 0-2 shall be included in this command and the DWord Length field set to account for these additional DWords.</td> <td>[Extended Parameter Enable] == 'true'</td> </tr> </tbody> </table>			Format:	=n	Value	Name	Programming Notes	Exists If	5h	5		[Extended Parameter Enable] == 'false'	8h	[Default]	When Extended Parameter Enable is true, three Dwords containing Extended Parameter 0-2 shall be included in this command and the DWord Length field set to account for these additional DWords.	[Extended Parameter Enable] == 'true'
Format:	=n																	
Value	Name	Programming Notes	Exists If															
5h	5		[Extended Parameter Enable] == 'false'															
8h	[Default]	When Extended Parameter Enable is true, three Dwords containing Extended Parameter 0-2 shall be included in this command and the DWord Length field set to account for these additional DWords.	[Extended Parameter Enable] == 'true'															
1	31:10	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>			Format:	MBZ												
Format:	MBZ																	

3DPRIMITIVE										
	<p>9 End Offset Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>If set, the Vertex Count Per Instance field is IGNORED, and the 3DPRIM_END_OFFSET register is used to indirectly specify the vertex count by defining the amount of valid data in VBO. The following restrictions apply:</p> <ul style="list-style-type: none"> • VBO must be enabled for use • VertexAccessType = SEQUENTIAL • Start Vertex Location = 0 • Start Instance Location = 0 • Base Vertex Location = 0 <p>Vertices are output until EndOffset is reached or exceeded in VBO. If EndOffset is reached or exceeded within the data associated with a vertex, that vertex is considered incomplete and will not be output. Partial objects will be discarded (as is normally done).</p> <p>If clear, End Offset is ignored.</p> <p>Indirect Parameter Enable and End Offset Enable must not be ENABLED at the same time, or behavior is UNDEFINED.</p>	Format:	Enable							
	Format:	Enable								
	<p>8 Vertex Access Type</p> <p>This field specifies how data held in vertex buffers marked as VERTEXDATA is accessed by Vertex Fetch.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>SEQUENTIAL</td> <td>VERTEXDATA buffers are accessed sequentially. Require if End Offset Enable is ENABLED.</td> </tr> <tr> <td>1h</td> <td>RANDOM</td> <td>VERTEXDATA buffers are accessed randomly via an index obtained from the Index Buffer.</td> </tr> </tbody> </table>	Value	Name	Description	0h	SEQUENTIAL	VERTEXDATA buffers are accessed sequentially. Require if End Offset Enable is ENABLED.	1h	RANDOM	VERTEXDATA buffers are accessed randomly via an index obtained from the Index Buffer.
	Value	Name	Description							
0h	SEQUENTIAL	VERTEXDATA buffers are accessed sequentially. Require if End Offset Enable is ENABLED.								
1h	RANDOM	VERTEXDATA buffers are accessed randomly via an index obtained from the Index Buffer.								
<p>7:6 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ									
<p>5:0 Primitive Topology Type</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Format:</td> <td>3D_Prim_Topo_Type</td> </tr> </table> <p>This field specifies the topology type of 3D primitive generated by this command. Note that a single primitive topology (list/strip/fan/etc.) can contain a number of basic objects (lines, triangles, etc.).</p> <p>This field is ignored. The topology type is specified via the 3DSTATE_VF_TOPOLOGY command.</p>	Format:	3D_Prim_Topo_Type								
Format:	3D_Prim_Topo_Type									
2	<p>31:0 Vertex Count Per Instance</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U32</td> </tr> </table> <p>This field specifies how many vertices are to be generated for each instance of the primitive topology. If End Offset Enable is clear: Format = U32 count of vertices Range =</p>	Format:	U32							
Format:	U32									

3DPRIMITIVE							
	<p>[0, 2³²-1] (upper limit probably constrained by VB size) Ignored if End Offset Enable or Indirect Parameter Enable is ENABLED.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2"> <ul style="list-style-type: none"> This per-instance value should specify a valid number of vertices for the primitive topology type. E.g., for 3DPRIM_TRILIST_ADJ, this field should specify a multiple of 6 vertices. However, in cases where too few or too many vertices are provided, the unused vertices will be silently discarded by the pipeline. A 0 value in this field effectively makes the command a 'no-operation'. </td> </tr> </tbody> </table>	Programming Notes		<ul style="list-style-type: none"> This per-instance value should specify a valid number of vertices for the primitive topology type. E.g., for 3DPRIM_TRILIST_ADJ, this field should specify a multiple of 6 vertices. However, in cases where too few or too many vertices are provided, the unused vertices will be silently discarded by the pipeline. A 0 value in this field effectively makes the command a 'no-operation'. 			
Programming Notes							
<ul style="list-style-type: none"> This per-instance value should specify a valid number of vertices for the primitive topology type. E.g., for 3DPRIM_TRILIST_ADJ, this field should specify a multiple of 6 vertices. However, in cases where too few or too many vertices are provided, the unused vertices will be silently discarded by the pipeline. A 0 value in this field effectively makes the command a 'no-operation'. 							
3	<p>31:0 Start Vertex Location</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">U32</td> </tr> </table> <p>This field specifies the "starting vertex" for each instance. This allows skipping over part of the vertices in a buffer if, for example, a previous 3DPRIMITIVE command had already drawn the primitives associated with the earlier entries. For SEQUENTIAL access, this field specifies, for each instance, a starting structure index into the vertex buffers. For RANDOM access, this field specifies, for each instance, a starting index into the Index Buffer.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2"> <ul style="list-style-type: none"> Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). Must be set to 0 if End Offset Enable is ENABLED. Ignored if Indirect Parameter Enable is ENABLED </td> </tr> </tbody> </table>	Format:	U32	Programming Notes		<ul style="list-style-type: none"> Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). Must be set to 0 if End Offset Enable is ENABLED. Ignored if Indirect Parameter Enable is ENABLED 	
Format:	U32						
Programming Notes							
<ul style="list-style-type: none"> Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). Must be set to 0 if End Offset Enable is ENABLED. Ignored if Indirect Parameter Enable is ENABLED 							
4	<p>31:0 Instance Count</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">U32</td> </tr> </table> <p>This field specifies the number of instances by which the primitive topology is to be regenerated. A value of 0 indicates "no instances" (no-op operation). A value of 1 effectively specifies "non-instanced" operation, though vertex buffers will still be used to provide instance data, if so programmed. Ignored if Indirect Parameter Enable is ENABLED.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0, FFFFFFFFh]</td> <td></td> </tr> </tbody> </table>	Format:	U32	Value	Name	[0, FFFFFFFFh]	
Format:	U32						
Value	Name						
[0, FFFFFFFFh]							
5	<p>31:0 Start Instance Location</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">U32</td> </tr> </table> <p>This field specifies the "starting instance" for the command as an initial structure index into Vertex Buffers for vertex elements with Instancing Enable set.</p> <p>Subsequent instances will access sequential instance data structures, as controlled by the Instance Data Step Rate.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2" style="height: 20px;"></td> </tr> </tbody> </table>	Format:	U32	Programming Notes			
Format:	U32						
Programming Notes							

3DPRIMITIVE					
		<ul style="list-style-type: none"> • Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). • Must be set to 0 if End Offset Enable is ENABLED. • Ignored if Indirect Parameter Enable is ENABLED. 			
6	31:0	<p>Base Vertex Location</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S31</td> </tr> </table> <p>This field specifies a signed bias to be added to values read from the index buffer. This allows the same index buffer values to access different vertex data for different commands. This field applies only to RANDOM access mode. This field is ignored for SEQUENTIAL access mode, where there Start Vertex Location can be used to specify different regions in the vertex buffers.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </table> <ul style="list-style-type: none"> • Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). • Must be set to 0 if End Offset Enable is ENABLED. • Ignored if Indirect Parameter Enable is ENABLED. 	Format:	S31	Programming Notes
Format:	S31				
Programming Notes					
7 Exists if: [Extended Parameters Present] == TRUE	31:0	<p>Extended Parameter 0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U32</td> </tr> </table> <p>If Indirect Parameter Enable is not set, this field specifies a U32 XP0 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.</p>	Format:	U32	
Format:	U32				
8 Exists if: [Extended Parameters Present] == TRUE	31:0	<p>Extended Parameter 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U32</td> </tr> </table> <p>If Indirect Parameter Enable is not set, this field specifies a U32 XP1 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.</p>	Format:	U32	
Format:	U32				
9 Exists if: [Extended Parameters Present] == TRUE	31:0	<p>Extended Parameter 2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U32</td> </tr> </table> <p>If Indirect Parameter Enable is not set, this field specifies a U32 XP2 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.</p>	Format:	U32	
Format:	U32				

3DSTATE_3D_MODE

3DSTATE_3D_MODE											
Source:	RenderCS										
Length Bias:	2										
This command is general 3D programming state that can be shared from the top to bottom of the pipeline.											
DWord	Bit	Description									
0	31:29	Command Type									
		Default Value: 3h GFXPIPE Format: OpCode									
	28:27	Command SubType									
		Default Value: 3h GFXPIPE_3D Format: OpCode									
	26:24	3D Command Opcode									
		Default Value: 1h GFXPIPE_NONPIPELINED Format: OpCode									
	23:16	3D Command Sub Opcode									
Default Value: 1Eh 3DSTATE_3D_MODE Format: OpCode											
15:8	Reserved										
	Format: MBZ										
7:0	DWord Length										
	Default Value: 1h Excludes DWord (0,1) Format: =n										
1	31:16	Mask Bits 1									
		Format: Enable[16] This field is the mask bits for the state bits below. This is a bit wise mask where the bit number-16 is the value of the corresponding bit being masked in the same data word. For example, if you want to update state for bits 3:2 then bits 19:18 must be set.									
	15:10	Reserved									
	Format: MBZ										
9	Float Blend Optimization Disable	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>[Default]</td> <td>Enables blend optimization for floating point RTs.</td> </tr> <tr> <td>1h</td> <td></td> <td>Disables blend optimization for floating point RTs.</td> </tr> </tbody> </table>	Value	Name	Description	0h	[Default]	Enables blend optimization for floating point RTs.	1h		Disables blend optimization for floating point RTs.
		Value	Name	Description							
		0h	[Default]	Enables blend optimization for floating point RTs.							
	1h		Disables blend optimization for floating point RTs.								
8	Reserved										

3DSTATE_3D_MODE

		Format:	MBZ
7	Reserved	Format:	MBZ
6	Slice Hashing Table Enable	Format:	Enable
	This field enables the use of indirect state SLICE_HASH_TABLE programmed via 3DSTATE_SLICE_HASH_STATE_POINTER .		
	Value	Name	Description
	0h	Disable [Default]	Slice Hashing is based on default lookup tables with the following function (X+Y) % total_enabled_subslices % enabled_slices
	1h	Enable	Slice Hashing is via SLICE_HASH_TABLE from 3DSTATE_SLICE_HASH_STATE_POINTER[total_enabled_subslices-4]
5	Subslice Hashing Table Enable	Format:	Enable
	This field enables the use of the Subslice Hashing Mode table programmed via 3DSTATE_SUBSLICE_HASH_TABLE .		
	Value	Name	Description
	0h	Disable [Default]	Subslice Hashing is computed
	1h	Enable	Subslice Hashing is via 3DSTATE_SUBSLICE_HASH_TABLE
4	3D Scoreboard Hashing Mode	Value	Name
	0	Scoreboard address calculation optimized for Subslice Hashing Mode [Default]	Before scoreboard address calculations one address bit will be removed to increase the efficiency of the scoreboard.
	1	Scoreboard address calculation not optimized	All address bits used when calculating scoreboard address.
	Programming Notes		
	When Subslice Hashing Table Enable and table entries do not contain a checkerboard pattern for each subslice, then optimizing for Subslice Hashing Mode may result in reduced performance due to increased false dependencies.		
3:2	Subslice Hashing Mode	This field is not used by the hardware. Hardware performs 16x16 hashing only.	

1:0	Cross Slice Hashing Mode		
	Format:		U2
	Value	Name	Description
	0h	Normal Mode [Default]	num_slices > 1 : 16x16 Hashing enabled num_slices == 1: No cross slice hashing
	1h	Cross Slice Hashing Disable	Disables the cross slice hashing
	2h	Reserved	Reserved
3h	32X32 hashing	32X32 pixel hashing across slices	This setting must be used when sub-slice hashing mode is 16x16 and num_slices > 1
Programming Notes			
A stalling flush is required before changing the value of this field. This is to make sure the entire pipeline is drained.			
2	31:16 Mask Bits 2		
	Format:		Enable[16]
	This field is the mask bits for the state bits below. This is a bit wise mask where the bit number-16 is the value of the corresponding bit being masked in the same data word. For example, if you want to update state for bits 6:0 then bits 22:16 must be set.		
15:10	Reserved		
	Format:		MBZ
9:7	Reserved		
	Format:		MBZ
6:0	AMFS MOCS		
	Format:		MEMORY_OBJECT_CONTROL_STATE
	Programming Notes		
SW must explicitly program AMFS_FLUSH using PIPE_CONTROL command prior to changing this field. SW must also explicitly flush DC cache flush if the setting of this field is changing from cacheable to non-cacheable in L3.			



3DSTATE_AA_LINE_PARAMETERS

DWord		Bit	Description
3DSTATE_AA_LINE_PARAMETERS			
Source:		RenderCS	
Length Bias:		2	
<p>The 3DSTATE_AA_LINE_PARAMS command is used to specify the slope and bias terms used in the improved alpha coverage computation (specifically for DX WHQL compliance). Note that in these devices the coverage values passed to PS threads are full U0.8 values, versus where U0.4 values are passed.</p>			
Workaround			
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,			
0	31:29	Command Type Default Value: 3h GFXPIPE Format: OpCode	
	28:27	Command SubType Default Value: 3h GFXPIPE_3D Format: OpCode	
	26:24	3D Command Opcode Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode	
	23:16	3D Command Sub Opcode Default Value: 0Ah 3DSTATE_AA_LINE_PARAMETERS Format: OpCode	
	15:8	Reserved Format: MBZ	
	7:0	Dword Length Default Value: 1h Excludes Dword (0,1) Format: =n	
1	31:24	AA Point Coverage Bias Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.	
	23:16	AA Coverage Bias Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.	
	15:8	AA Point Coverage Slope	

3DSTATE_AA_LINE_PARAMETERS

		<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U0.8</td> </tr> </table> <p>This field specifies the slope term to be used in the aa coverage computation for edges 0 and 3. If this field is zero, the Windower will revert to legacy aa line coverage computation (though still output expanded U0.8 coverage values).</p>	Format:	U0.8
Format:	U0.8			
	7:0	<p>AA Coverage Slope</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U0.8</td> </tr> </table> <p>This field specifies the slope term to be used in the aa coverage computation for edges 0 and 3. If this field is zero, the Windower will revert to legacy aa line coverage computation (though still output expanded U0.8 coverage values).</p>	Format:	U0.8
Format:	U0.8			
2	31:24	<p>AA Point Coverage EndCap Bias</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U0.8</td> </tr> </table> <p>This field specifies the bias term to be used in the aa coverage computation for edges 1 and 2.</p>	Format:	U0.8
	Format:	U0.8		
	23:16	<p>AA Coverage EndCap Bias</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U0.8</td> </tr> </table> <p>This field specifies the bias term to be used in the aa coverage computation for edges 1 and 2.</p>	Format:	U0.8
	Format:	U0.8		
15:8	<p>AA Point Coverage EndCap Slope</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U0.8</td> </tr> </table> <p>This field specifies the slope term to be used in the aa coverage computation for edges 1 and 2.</p>	Format:	U0.8	
Format:	U0.8			
7:0	<p>AA Coverage EndCap Slope</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U0.8</td> </tr> </table> <p>This field specifies the slope term to be used in the aa coverage computation for edges 1 and 2.</p>	Format:	U0.8	
Format:	U0.8			



3DSTATE_AMFS

3DSTATE_AMFS		
Source:	BSpec	
Length Bias:	2	
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: Opcode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
		Format: Opcode
	26:24	3D Command Opcode
Default Value: 0h 3DSTATE_PIPELINED		
Format: Opcode		
23:16	3D Command Sub Opcode	
	Default Value: 6Fh 3DSTATE_AMFS	
	Format: Opcode	
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	Format: =n
		n = Total Length -2
	Value	Name
	0h	Excludes DWord (0,1)
1	31:0	AMFS State Body
		Format: 3DSTATE_AMFS_BODY

3DSTATE_AMFS_TEXTURE_POINTERS

3DSTATE_AMFS_TEXTURE_POINTERS			
Source:	BSpec		
Length Bias:	2		
<p>Specifies bindless surfacestate offsets for procedural texture slots <0,7>. A Procedural Texture slot declared here matches a Texel Shader kernel slot declared in 3DSTATE_AMFS command. All procedural texture slots must be programmed:</p> <ul style="list-style-type: none"> - To enable a procedural texture slot: program the slots BindlessSurfaceOffset to an offset that corresponds to a surface with the ProceduralTexture bit set that will be used in an <i>evaluate</i> message by an invoking shader. - To disable a procedural texture slot: program the slots BindlessSurfaceOffset to an offset that is either known not to be used with evaluate messages, or that is known to correspond to a surface that does not have the ProceduralTexture bit set 			
DWord	Bit	Description	
0	31:29	Type	
		Default Value:	3h GFXPIPE
		Format:	Opcode
	28:27	SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	Opcode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		72h 3DSTATE_AMFS_TEXTURE_POINTERS	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	9 Excludes DWord (0,1)	
	Format:	=n	
	n = Total Length - 2		
1..10	319:0	AMFS Texture Pointers State Body	
		Format: 3DSTATE_AMFS_TEXTURE_POINTERS_BODY	



3DSTATE_BINDING_TABLE_POINTERS_DS

3DSTATE_BINDING_TABLE_POINTERS_DS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_BINDING_TABLE_POINTERS_DS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	28h 3DSTATE_BINDING_TABLE_POINTERS_DS
		Format:	OpCode
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Binding Table Pointers State Body	
		Format:	3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_GS

3DSTATE_BINDING_TABLE_POINTERS_GS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_BINDING_TABLE_POINTERS_GS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	29h 3DSTATE_BINDING_TABLE_POINTERS_GS
		Format:	OpCode
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Binding Table Pointers State Body	
		Format:	3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_HS

3DSTATE_BINDING_TABLE_POINTERS_HS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_BINDING_TABLE_POINTERS_HS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	27h 3DSTATE_BINDING_TABLE_POINTERS_HS
		Format:	OpCode
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Binding Table Pointers State Body	
		Format:	3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_PS

3DSTATE_BINDING_TABLE_POINTERS_PS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_BINDING_TABLE_POINTERS_PS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		2Ah 3DSTATE_BINDING_TABLE_POINTERS_PS	
Format:		OpCode	
15	POSH-Enabled Render State Optimization Enable		
14:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Binding Table Pointers State Body	
		Format:	3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_VS

3DSTATE_BINDING_TABLE_POINTERS_VS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_BINDING_TABLE_POINTERS_VS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	26h 3DSTATE_BINDING_TABLE_POINTERS_VS
		Format:	OpCode
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Binding Table Pointers State Body	
		Format:	3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POOL_ALLOC

DWord		Bit	Description
Source:		RenderCS	
Length Bias:		2	
<p>This command is to program the base address and size of the binding table pool. The address to fetch the binding table is based on the Binding Table Pool Base Address and the binding table pointer if the Binding Table Pool is enabled. Otherwise the binding table pointer is an offset from the Surface Base Address.</p>			
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
	26:24	3D Command Opcode	
		Default Value:	1h 3DSTATE_NONPIPELINED
23:16	3D Command Sub Opcode		
	Default Value:	19h 3DSTATE_BINDING_TABLE_POOL_ALLOC	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	2h DWORD_COUNT_n	
	Format:		=n
	1..2	63:12	Binding Table Pool Base Address
Format:			VIRTUAL_ADDR[63:12]
<p>Specifies the 4K-byte aligned base address for Binding Table Pool. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</p>			
	11	Binding Table Pool Enable	
		Format:	U1
<p>When this bit is set it enables HW generation of binding tables. When this bit is cleared it disables HW generation of binding tables.</p>			
Programming Notes			
<p>If the Binding Table Pool Enable is set when resource streamer is disabled, the engine will fetch binding table entries from the Binding Table Pool Address. In this case, driver must ensure the</p>			

3DSTATE_BINDING_TABLE_POOL_ALLOC											
		pool is populated and the binding table is coherent prior to engine fetching binding table entries. The format of the binding table must match the same format output by the resource streamer hardware.									
	10:7	Reserved Format: _____ MBZ									
	6:0	Surface Object Control State Format: _____ MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface. <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Programming Notes</div> Bit 0 is not programmable and is always zero.									
3	31:12	Binding Table Pool Buffer Size Format: _____ U20 This field specifies the size of the buffer in 4K pages. Any access which straddle or go past the end of the buffer will return 0.									
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,1048575]</td> <td></td> <td></td> </tr> <tr> <td>0</td> <td>No Valid Data</td> <td>There is no valid data in the buffer</td> </tr> </tbody> </table>	Value	Name	Description	[0,1048575]			0	No Valid Data	There is no valid data in the buffer
		Value	Name	Description							
		[0,1048575]									
		0	No Valid Data	There is no valid data in the buffer							
		<div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Restriction</div> Programming size of zero is illegal in the case that the pool is enabled.									
11:0	Reserved Format: _____ MBZ										

3DSTATE_BLEND_STATE_POINTERS

3DSTATE_BLEND_STATE_POINTERS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_BLEND_STATE_POINTERS command is used to set up the pointers to the color calculator state.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		24h 3DSTATE_BLEND_STATE_POINTERS	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Blend State Pointer State Body	
		Format:	3DSTATE_BLEND_STATE_POINTERS_BODY



3DSTATE_CC_STATE_POINTERS

3DSTATE_CC_STATE_POINTERS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_CC_STATE_POINTERS command is used to set up the pointers to the color calculator state.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	0Eh 3DSTATE_CC_STATE_POINTERS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	CC State Pointers Body	
		Format:	3DSTATE_CC_STATE_POINTERS_BODY

3DSTATE_CHROMA_KEY

3DSTATE_CHROMA_KEY					
Source:	RenderCS				
Length Bias:	2				
<p>The 3DSTATE_CHROMA_KEY instruction is used to program texture color/chroma-key key values. A table containing four set of values is supported. The ChromaKey Index sampler state variable is used to select which table entry is associated with the map. Texture chromakey functions are enabled and controlled via use of the ChromaKey Enable texture sampler state variable. Texture Color Key (keying on a paletted texture index) is not supported.</p>					
DWord	Bit	Description			
0	31:29	Command Type			
		Default Value: 3h GFXPIPE Format: Opcode			
	28:27	Command SubType			
		Default Value: 3h GFXPIPE_3D Format: Opcode			
	26:24	3D Command Opcode			
		Default Value: 1h 3DSTATE_NONPIPELINED Format: Opcode			
	23:16	3D Command Sub Opcode			
Default Value: 04h 3DSTATE_CHROMA_KEY Format: Opcode					
15:8	Reserved				
7:0	Format: MBZ				
	DWord Length				
Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2					
1	31:30	ChromaKey Table Index			
		Format: U2 Selects which entry in the ChromaKey table is to be loaded			
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,3]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,3]
Value	Name				
[0,3]					
29:0	Reserved				
Format: MBZ					
2	31:0	ChromaKey Low Value			

3DSTATE_CHROMA_KEY																																																								
	<p>This field specifies the "low" (minimum) value of the chroma key range. Texel samples are considered "matching the key" if each component of the texel falls within the (inclusive) chroma range. See ChromaKey High Value for further format, programming info.</p>																																																							
3	<p>31:0 ChromaKey High Value This field specifies the "high" (maximum) value of the chroma key range. Texel samples are considered "matching the key" if each component of the texel falls within the (inclusive) chroma range.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="5" style="text-align: center; background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="5">ChromaKey values are specified using 8-bit channels. When using surface formats with less than 8 bits per channel, the device will expand channels by replicating the required number of MSBs into the LSBs of each channel. Software must account for this conversion when it programs Chromakey Low/High Values (e.g., by performing the same replication).</td> </tr> <tr> <td colspan="5">For channels that do not exist in the actual surface (e.g., Alpha channel for non-ARGB maps), software must explicitly program full range high/low values (High=FFh, Low=0h for formats using unsigned chroma key values, High=7Fh, Low=FFh for formats using sign magnitude chroma key values) in order to effectively remove the comparison of that field from the ChromaKey function.</td> </tr> <tr> <td colspan="5">For channels in SNORM format in the surface format, the value in the high/low value for that channel is interpreted in sign magnitude format. Negative zero value is not supported (use positive zero instead). For channels with mixed UNORM/SNORM formats (i.e. R5G5_SNORM_B6_UNORM), the ChromaKey is programmed as if all channels are SNORM.</td> </tr> <tr> <td colspan="5">YUV ChromaKey will use an interpolated chrominance value from the map for comparison to the chroma key values for those texels without chrominance due to downsampling. The chrominance value used is the average of values to the left and right of the texel in question.</td> </tr> <tr> <td colspan="5">It is UNDEFINED to program any component of the ChromaKey High Value to be less than the corresponding component of ChromaKey Low Value.</td> </tr> <tr> <td colspan="5">Format = interpreted according to associated texel format "class":</td> </tr> <tr> <td colspan="5">Only the surface formats listed as supported for chroma key in the surface formats table can be used with this feature. Use of any other surface format with chroma key enabled is UNDEFINED.</td> </tr> <tr> <th style="text-align: center;">Surface Format</th> <th style="text-align: center;">31:24</th> <th style="text-align: center;">23:15</th> <th style="text-align: center;">16:8</th> <th style="text-align: center;">7:0</th> </tr> <tr> <td>ARGB and BC (DXT) formats</td> <td style="text-align: center;">A</td> <td style="text-align: center;">R</td> <td style="text-align: center;">G</td> <td style="text-align: center;">B</td> </tr> <tr> <td>YCrCb formats</td> <td style="text-align: center;">A</td> <td style="text-align: center;">Cr</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Cb</td> </tr> </tbody> </table>	Programming Notes					ChromaKey values are specified using 8-bit channels. When using surface formats with less than 8 bits per channel, the device will expand channels by replicating the required number of MSBs into the LSBs of each channel. Software must account for this conversion when it programs Chromakey Low/High Values (e.g., by performing the same replication).					For channels that do not exist in the actual surface (e.g., Alpha channel for non-ARGB maps), software must explicitly program full range high/low values (High=FFh, Low=0h for formats using unsigned chroma key values, High=7Fh, Low=FFh for formats using sign magnitude chroma key values) in order to effectively remove the comparison of that field from the ChromaKey function.					For channels in SNORM format in the surface format, the value in the high/low value for that channel is interpreted in sign magnitude format. Negative zero value is not supported (use positive zero instead). For channels with mixed UNORM/SNORM formats (i.e. R5G5_SNORM_B6_UNORM), the ChromaKey is programmed as if all channels are SNORM.					YUV ChromaKey will use an interpolated chrominance value from the map for comparison to the chroma key values for those texels without chrominance due to downsampling. The chrominance value used is the average of values to the left and right of the texel in question.					It is UNDEFINED to program any component of the ChromaKey High Value to be less than the corresponding component of ChromaKey Low Value.					Format = interpreted according to associated texel format "class":					Only the surface formats listed as supported for chroma key in the surface formats table can be used with this feature. Use of any other surface format with chroma key enabled is UNDEFINED.					Surface Format	31:24	23:15	16:8	7:0	ARGB and BC (DXT) formats	A	R	G	B	YCrCb formats	A	Cr	Y	Cb
Programming Notes																																																								
ChromaKey values are specified using 8-bit channels. When using surface formats with less than 8 bits per channel, the device will expand channels by replicating the required number of MSBs into the LSBs of each channel. Software must account for this conversion when it programs Chromakey Low/High Values (e.g., by performing the same replication).																																																								
For channels that do not exist in the actual surface (e.g., Alpha channel for non-ARGB maps), software must explicitly program full range high/low values (High=FFh, Low=0h for formats using unsigned chroma key values, High=7Fh, Low=FFh for formats using sign magnitude chroma key values) in order to effectively remove the comparison of that field from the ChromaKey function.																																																								
For channels in SNORM format in the surface format, the value in the high/low value for that channel is interpreted in sign magnitude format. Negative zero value is not supported (use positive zero instead). For channels with mixed UNORM/SNORM formats (i.e. R5G5_SNORM_B6_UNORM), the ChromaKey is programmed as if all channels are SNORM.																																																								
YUV ChromaKey will use an interpolated chrominance value from the map for comparison to the chroma key values for those texels without chrominance due to downsampling. The chrominance value used is the average of values to the left and right of the texel in question.																																																								
It is UNDEFINED to program any component of the ChromaKey High Value to be less than the corresponding component of ChromaKey Low Value.																																																								
Format = interpreted according to associated texel format "class":																																																								
Only the surface formats listed as supported for chroma key in the surface formats table can be used with this feature. Use of any other surface format with chroma key enabled is UNDEFINED.																																																								
Surface Format	31:24	23:15	16:8	7:0																																																				
ARGB and BC (DXT) formats	A	R	G	B																																																				
YCrCb formats	A	Cr	Y	Cb																																																				

3DSTATE_CLEAR_PARAMS

3DSTATE_CLEAR_PARAMS		
Source:	RenderCS	
Length Bias:	2	
<p>This command defines the depth clear value delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).</p> <p>HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
Default Value: 04h 3DSTATE_CLEAR_PARAMS Format: OpCode		
15:8	Reserved	
	Format: MBZ	
7:0	Dword Length	
	Default Value: 1h Excludes Dword (0,1) Format: =n	
1..2	63:0	Clear Params State Body
		Format: 3DSTATE_CLEAR_PARAMS_BODY

3DSTATE_CLIP

3DSTATE_CLIP			
Source:	RenderCS		
Length Bias:	2		
Restriction			
When executed in POCS command stream, this programs the state for CLR stage of the POCS pipeline			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	12h 3DSTATE_CLIP
		Format:	OpCode
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	02h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..3	95:0	Clip State Body	
		Format:	3DSTATE_CLIP_BODY

3DSTATE_CONSTANT_ALL

3DSTATE_CONSTANT_ALL			
Source:	RenderCS, PositionCS		
Length Bias:	2		
This instruction species pointers and sizes of data to be fetched from memory and loaded as part of the shaders thread payload.			
Programming Notes			
The programming of enabled buffers is in ascending order where if buffer zero is programmed it will be the first pointer programmed and parsed. Any buffers being omitted must not be programmed.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	6Dh 3DSTATE_CONSTANT_ALL
		Format:	OpCode
	15	POSH-Enabled Render State Optimization Enable	
		<p>This bit allows a state command to participate in the POSH-Enabled Render State Optimization function. That function skips over selected commands deemed unnecessary due to the subsequent POSH-Enabled 3DPRIMITIVE command being completely culled by the POSH pipeline.</p> <p>Software shall ensure that the following conditions are met as a prerequisite to setting this bit:</p> <ol style="list-style-type: none"> 1. The state command is only executed by the Render pipeline. (The bit must be cleared if the state command is executed by the POSH pipeline.) 2. The very next 3DPRIMITIVE command has POSH-Enabled specified. 3. The state programmed is <u>only used by the very next</u> 3DPRIMITIVE command. I.e., following the very next 3DPRIMITIVE command, the relevant state is reprogrammed prior to a subsequent 3DPRIMITIVE command. <p>For improved performance, Software should set this bit if these conditions are met, however Software is not required to do so.</p>	
	14:13	Reserved	
Format:		MBZ	

3DSTATE_CONSTANT_ALL														
	12:8	Shader Mask This bit specifies if the updated pointers are valid for specific shaders. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>xxxx1b</td> <td>Vertex Shader Update Enable</td> </tr> <tr> <td>xxx1xb</td> <td>Hull Shader Update Enable</td> </tr> <tr> <td>xx1xxb</td> <td>Domain Shader Update Enable</td> </tr> <tr> <td>x1xxxb</td> <td>Geometry Shader Update Enable</td> </tr> <tr> <td>1xxxxb</td> <td>Pixel Shader Update Enable</td> </tr> </tbody> </table>	Value	Name	xxxx1b	Vertex Shader Update Enable	xxx1xb	Hull Shader Update Enable	xx1xxb	Domain Shader Update Enable	x1xxxb	Geometry Shader Update Enable	1xxxxb	Pixel Shader Update Enable
	Value	Name												
	xxxx1b	Vertex Shader Update Enable												
	xxx1xb	Hull Shader Update Enable												
	xx1xxb	Domain Shader Update Enable												
	x1xxxb	Geometry Shader Update Enable												
	1xxxxb	Pixel Shader Update Enable												
7:0	DWord Length Format: _____ =n n = 2b (where b = # of pointers included) <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>DWORD_COUNT_n [Default]</td> </tr> <tr> <td>[0,8]</td> <td>0-4 Pointers</td> </tr> </tbody> </table>	Value	Name	8	DWORD_COUNT_n [Default]	[0,8]	0-4 Pointers							
Value	Name													
8	DWORD_COUNT_n [Default]													
[0,8]	0-4 Pointers													
1	31	Update Mode If set, pointers that are not valid will retain their value. If clear, then all pointers not valid will be cleared with zero address and zero size. If pointer is valid, then the value of this field is a don't care and the value programmed is always loaded. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Clear pointer and size of non-valid pointers</td> </tr> <tr> <td>1</td> <td>Retain value of non-valid pointers</td> </tr> </tbody> </table>	Value	Name	0	Clear pointer and size of non-valid pointers	1	Retain value of non-valid pointers						
	Value	Name												
	0	Clear pointer and size of non-valid pointers												
	1	Retain value of non-valid pointers												
	30:20	Reserved Format: _____ MBZ												
19:16	Pointer Buffer Mask This bit specifies which pointers are valid in this command. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>xxx1b</td> <td>Buffer 0 Valid</td> </tr> <tr> <td>xx1xb</td> <td>Buffer 1 Valid</td> </tr> <tr> <td>x1xxb</td> <td>Buffer 2 Valid</td> </tr> <tr> <td>1xxxb</td> <td>Buffer 3 Valid</td> </tr> </tbody> </table>	Value	Name	xxx1b	Buffer 0 Valid	xx1xb	Buffer 1 Valid	x1xxb	Buffer 2 Valid	1xxxb	Buffer 3 Valid			
Value	Name													
xxx1b	Buffer 0 Valid													
xx1xb	Buffer 1 Valid													
x1xxb	Buffer 2 Valid													
1xxxb	Buffer 3 Valid													
15:7	Reserved Format: _____ MBZ													
	6:0	Constant Buffer Object Control State Format: _____ MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.												
	2..n	Constant All Data Format: _____ 3DSTATE_CONSTANT_ALL_DATA[4]												

3DSTATE_CONSTANT_DS

3DSTATE_CONSTANT_DS		
Source:	RenderCS	
Length Bias:	2	
This command sets pointers to the push constants for the DS unit. The constant data pointed to by this command is loaded into the DS unit's push constant buffer (PCB).		
Workaround		
<p>Workaround :</p> <p>The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:</p> <ul style="list-style-type: none"> • always force buffer 3 to have a non zero read length • always force buffer 0 to a zero read length 		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
		Default Value: 1Ah 3DSTATE_CONSTANT_DS Format: OpCode
15	Reserved	
	Format: MBZ	
14:8	Constant Buffer Object Control State	
	Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.	
7:0	DWord Length	
	Default Value: 9h Excludes DWord (0,1) Format: =n	

3DSTATE_CONSTANT_DS				
1..10	319:0	<p>Constant Body</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>3DSTATE_CONSTANT(Body)</td> </tr> </table> <p>See the 3DSTATE_CONSTANT(Body) format for the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS.</p>	Format:	3DSTATE_CONSTANT(Body)
Format:	3DSTATE_CONSTANT(Body)			

3DSTATE_CONSTANT_GS

3DSTATE_CONSTANT_GS			
Source:	RenderCS		
Length Bias:	2		
This command sets pointers to the push constants for the GS unit. The constant data pointed to by this command will be loaded into the GS unit's push constant buffer (PCB).			
Workaround			
<p>Workaround :</p> <p>The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:</p> <ul style="list-style-type: none"> • always force buffer 3 to have a non zero read length • always force buffer 0 to a zero read length 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	16h 3DSTATE_CONSTANT_GS
		Format:	OpCode
	15	Reserved	
		Format:	MBZ
14:8	Constant Buffer Object Control State		
	Format:	MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.	
7:0	DWord Length		
	Default Value:	9h Excludes DWord (0,1)	
	Format:	=n	

3DSTATE_CONSTANT_GS				
1..10	319:0	<p>Constant Body</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>3DSTATE_CONSTANT(Body)</td> </tr> </table> <p>Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS</p>	Format:	3DSTATE_CONSTANT(Body)
Format:	3DSTATE_CONSTANT(Body)			

3DSTATE_CONSTANT_HS

DWord		Bit	Description
Source:		RenderCS	
Length Bias:		2	
<p>This command sets pointers to the push constants for the HS unit. The constant data pointed to by this command is loaded into the HS unit's push constant buffer (PCB).</p>			
Workaround			
<p>Workaround :</p> <p>The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:</p> <ul style="list-style-type: none"> • always force buffer 3 to have a non zero read length • always force buffer 0 to a zero read length 			
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	19h 3DSTATE_CONSTANT_HS
		Format:	OpCode
	15	Reserved	
		Format:	MBZ
14:8	Constant Buffer Object Control State		
	Format:	MEMORY_OBJECT_CONTROL_STATE	
Specifies the memory object control state for all constant buffers defined in this command.			
7:0	DWord Length		
	Default Value:	9h Excludes DWord (0,1)	
	Format:	=n	

3DSTATE_CONSTANT_HS				
1..10	319:0	<p>Constant Body</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>3DSTATE_CONSTANT(Body)</td> </tr> </table> <p>Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS</p>	Format:	3DSTATE_CONSTANT(Body)
Format:	3DSTATE_CONSTANT(Body)			

3DSTATE_CONSTANT_PS

3DSTATE_CONSTANT_PS		
Source:	RenderCS	
Length Bias:	2	
This command sets pointers to the push constants for the PS unit. The constant data pointed to by this command is loaded into the PS unit's push constant buffer (PCB).		
Programming Notes		
A 3DSTATE_GATHER_PS command must be dispatched along with any 3DSTATE_CONSTANT_PS command when the Gather Pool is enabled within a batch buffer.		
The 3DSTATE_CONSTANT_* command is not committed to the shader unit until the corresponding (same shader) 3DSTATE_BINDING_TABLE_POINTER_* command is parsed. For example, the 3DSTATE_CONSTANT_VS command will not fetch the constant buffers from memory and make available to the shader until the 3DSTATE_BINDING_TABLE_POINTERS_VS is parsed by the render command streamer. In case of multiple 3DSTATE_CONSTANT_VS programmed prior to 3DSTATE_BINDING_TABLE_POINTER_VS, only the most recently programmed 3DSTATE_CONSTANT_VS will be committed. On usage model of enabling legacy mode is when Resource Streamer is not enabled.		
Workaround		
Workaround : The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:		
<ul style="list-style-type: none"> • always force buffer 3 to have a non zero read length • always force buffer 0 to a zero read length 		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
	Format: OpCode	
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
	Format: OpCode	
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED
Format: OpCode		
23:16	3D Command Sub Opcode	
	Default Value: 17h 3DSTATE_CONSTANT_PS	

3DSTATE_CONSTANT_PS		
		Format: OpCode
	15	Reserved Format: MBZ
	14:8	Constant Buffer Object Control State Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	Dword Length Default Value: 9h Excludes DWord (0,1) Format: =n
1..10	319:0	Constant Body Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_TS_POINTER

3DSTATE_CONSTANT_TS_POINTER			
Source:	RenderCS		
Length Bias:	2		
<p>This command sets pointers to the Texel Shaders push constants for AMFS unit. The constant data pointed to by this command is stored in memory, not in push constant buffer (PCB).</p> <p>The 3DSTATE_CONSTANT_TS_POINTER command gets committed to the shader on parsing 3DPRIMITIVE command or on encountering an explicit/implicit flush (Texel Shaders do not use binding table or Resource Streamer).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		73h 3DSTATE_CONSTANT_TS_POINTER	
Format:		OpCode	
15	Reserved		
	Format:	MBZ	
14:8	Constant Buffer Object Control State		
	Format:	MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for constant buffers defined in this command.	
7:0	DWord Length		
	Default Value:	1h Excludes DWord (0,1)	
	Format:	=n	
	n = Total Length -2		
1..2	63:0	Constant TS Pointer State Body	
		Format:	3DSTATE_CONSTANT_TS_POINTER_BODY

3DSTATE_CONSTANT_VS

3DSTATE_CONSTANT_VS			
Source:	RenderCS		
Length Bias:	2		
This command sets pointers to the push constants for VS unit. The constant data pointed to by this command is loaded into the VS unit's push constant buffer (PCB).			
Programming Notes			
When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline.			
Workaround			
<p>Workaround :</p> <p>The driver must ensure The following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:</p> <ul style="list-style-type: none"> • always force buffer 3 to have a non zero read length • always force buffer 0 to a zero read length 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	15h 3DSTATE_CONSTANT_VS
		Format:	OpCode
	15	Reserved	
		Format:	MBZ
14:8	Constant Buffer Object Control State		
	Format:	MEMORY_OBJECT_CONTROL_STATE	
Specifies the memory object control state for all constant buffers defined in this command.			

3DSTATE_CONSTANT_VS		
	7:0	DWord Length
		Default Value: 9h Excludes DWord (0,1)
		Format: =n
1..10	319:0	Constant Body
		Format: 3DSTATE_CONSTANT(Body)
<p>Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS</p>		



3DSTATE_CPS_POINTERS

3DSTATE_CPS_POINTERS			
Source:		RenderCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	Opcode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	Opcode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		22h 3DSTATE_CPS_POINTERS	
Format:		OpCode	
15:0	DWord Length		
	Default Value:	0h	
	Format:	=n	
1	31:0	CPS Pointers State Body	
		Format:	3DSTATE_CPS_POINTERS_BODY

3DSTATE_DEPTH_BOUNDS

3DSTATE_DEPTH_BOUNDS			
Source:		BSpec	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	03h GFXPIPE
		Format:	Opcode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	Opcode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	Opcode
	23:16	3D Command Sub Opcode	
Default Value:		71h 3DSTATE_DEPTH_BOUNDS	
Format:		Opcode	
15	Depth Bounds Test Enable Modify Disable		
	Format:	Disable	
When this bit is set, the following fields will be ignored:			
<ul style="list-style-type: none"> Depth Bounds Test Enable 			
14	Depth Bounds Test Value Modify Disable		
	Format:	Disable	
When this bit is set, the following fields will be ignored:			
<ul style="list-style-type: none"> Depth Bounds Test Min Value Depth Bounds Test Max Value 			
13:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	02h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..3	95:0	Depth Bounds State Body	
		Format:	3DSTATE_DEPTH_BOUNDS_BODY



3DSTATE_DEPTH_BUFFER

3DSTATE_DEPTH_BUFFER				
Source:	RenderCS			
Length Bias:	2			
<p>The depth buffer surface state is delivered as a pipelined state packet. However, the state change pipelining isn't completely transparent (see restriction below).</p> <p>WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.</p>				
Programming Notes				
<p>Note for validation teams. If the depth surface is backdoor initialized or written to directly by the CPU, the values placed in the Depth Surface must be within the numeric range of [0.0 ... 1.0] for DirectX and may in the future include +/- max floating point values; but not +/-Inf, DNORMs or any NaN code.</p> <p>If the Depth surface is not present, SW must set the Surface Type field to SURFTYPE_NULL</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value: 3h GFXPIPE Format: OpCode		
	28:27	Command SubType		
		Default Value: 3h GFXPIPE_3D Format: OpCode		
	26:24	3D Command Opcode		
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode		
	23:16	3D Command Sub Opcode		
Default Value: 5h 3DSTATE_DEPTH_BUFFER Format: OpCode				
15:8	Reserved			
	Format: MBZ			
7:0	DWord Length	Format: =n		
		Excludes DWord(0,1)		
	Value	Name		
	6h	Excludes Dword (0,1) [Default]		
1	31:29	Surface Type		
		Value	Name	Description
		0h	Reserved	Reserved

3DSTATE_DEPTH_BUFFER

		1h	SURFTYPE_2D	Defines a 2-dimensional map or array of maps
		2h	Reserved	Reserved
		3h	SURFTYPE_CUBE	Defines a cube map
		4h-6h	Reserved	
		7h	SURFTYPE_NULL	Defines a null surface
Programming Notes				
<p>The Surface Type of the depth buffer must be the same as the Surface Type of the</p> <ol style="list-style-type: none"> 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL. <p>If depth is enabled with 1D render target, depth surface type needs to be set to 2D surface type and height set to 1. For this case only, the Surface Type of the depth buffer can be 2D while the Surface Type of the render target(s) are 1D, representing an exception to a programming note above.</p>				
28	Depth Write Enable			
	Format:		Enable	
This field enables depth writes to the depth buffer surface. Both this field and the Depth Buffer Write Enable field in DEPTH_STENCIL_STATE must be enabled in order for depth writes to occur.				
27	Null Page Coherency Enable			
	Format:		Enable	
This field is used for enabling NULL coherency as defined under Tiled Resources.				
	Value	Name		
	1	Enable		
	0	Disable [Default]		
Programming Notes				
SW must enable this bit only if Tiled Resource is enabled				
26:24	Surface Format			
Specifies the format of the depth buffer.				
	Value	Name		
	0h	Reserved		
	1h	D32_FLOAT		
	2h	Reserved		
	3h	D24_UNORM_X8_UINT		
	4h	Reserved		
	5h	D16_UNORM		

3DSTATE_DEPTH_BUFFER

	6h-7h	Reserved											
23	Corner Texel Mode <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">Enable</td> </tr> </table> <p>This field, when ENABLED, indicates when a surface is using corner texel-mode for depth surface. This bit changes how the size of each MIP when calculating the offset within a surface.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">Disable [Default]</td> <td>Corner Texel mode is not enabled.</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">Enable</td> <td>Corner Texel Mode is enabled.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Corner texel for the depth buffer must be the same as the Surface Type of the</p> <ol style="list-style-type: none"> 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL 		Format:	Enable	Value	Name	Description	0h	Disable [Default]	Corner Texel mode is not enabled.	1h	Enable	Corner Texel Mode is enabled.
Format:	Enable												
Value	Name	Description											
0h	Disable [Default]	Corner Texel mode is not enabled.											
1h	Enable	Corner Texel Mode is enabled.											
22	Hierarchical Depth Buffer Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">Enable</td> </tr> </table> <p>If enabled, indicates that a hierarchical depth buffer is defined.</p> <p style="text-align: center;">Programming Notes</p> <p>If this field is enabled, the Software Tiled Rendering Mode must be NORMAL. This field must be disabled if Early Depth Test Enable is disabled OR if depth buffer surface type is NULL. This field must be disabled if surface type field is SURFTYPE_1D</p>		Format:	Enable									
Format:	Enable												
21	Depth Buffer Compression Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">Enable</td> </tr> </table> <p>if enabled, indicates that Depth Buffer Compression is Enabled</p> <p>When this field is enabled, Depth Buffer must be initialized via Depth Clear (HZ_OP) when HiZ is enabled. If HiZ is disabled, Depth Buffer must be initialized via full screen primitive with Depth Write enabled and Depth Test Disabled.</p> <p style="text-align: center;">Programming Notes</p> <p>SW must set this bit if the Depth Control surface enable is also set. The depth surface control enable is in Bit[19] of this DWORD.</p>		Format:	Enable									
Format:	Enable												
20	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>		Format:	MBZ									
Format:	MBZ												
19	Control Surface Enable <p>If set to 1, it indicates if the common control surface is present. The read and write transaction opcodes sent by RCZto the fabric are different depending on the control surface. If the control surface is not present, the reads and writes are in legacy mode. If the control surface is present, the reads and write opcodes will be either UNCOMPRESSED_TYP for uncompressible transactions (resolves) or COMPRESSED_TYP for compressible transactions.</p> <p style="text-align: center;">Programming Notes</p>												

3DSTATE_DEPTH_BUFFER													
	SW must set this bit to "1", if the common control surface is present in the system.												
18	Reserved Format: MBZ												
17:0	Surface Pitch Format: U18-1 For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to <i>Memory Data Formats</i> section for the definition of the Cu parameter depending on the case). The range in bytes is $[2^{Cu}-1, 262143]$ -> $[(2^{Cu})B, 256KB] = [1 \text{ tile}, 256KB/(2^{Cu}) \text{ tiles}]$ <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>[7Fh, 3FFFFh]</td> <td></td> <td>corresponding to [128B, 256KB] also restricted to a multiple of 128B</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>The minimum pitch should be calculated based on Cu, Cv, W_i. The Cu, Cv are the tile constants and W_i is the aligned width adjusted for MSAA. Refer to "2D Surfaces" to get the Cu, Cv, W_i values and Calculations. Then use this for pitch formula : $Minimum_pitch = (\text{ceiling}(W_0 * \text{pixel_size}) / (1 \ll Cu)) * (1 \ll Cv) - 1$; //W₀ is the aligned width for the largest LOD (i.e LOD 0) $(1 \ll Cu) = \text{tile width in bytes}$ $(1 \ll Cv) = \text{tile height in lines}$</p>	Value	Name	Description	[7Fh, 3FFFFh]		corresponding to [128B, 256KB] also restricted to a multiple of 128B						
Value	Name	Description											
[7Fh, 3FFFFh]		corresponding to [128B, 256KB] also restricted to a multiple of 128B											
2..3	63:0 Surface Base Address Format: VIRTUAL_ADDR[63:0] <p style="text-align: center;">Programming Notes</p> This field specifies address of the buffer in mapped Graphics Memory. The Depth Buffer can only be mapped to Main Memory (uncached). If the surface is tiled, the base address must conform to the Per-Surface Tiling Alignment. If the buffer is linear, the surface must be 64-byte aligned. If the buffer is linear, the surface must be 64-byte aligned.												
4	31 Reserved Format: MBZ												
	30:17 Height Format: U14-1 This field specifies the height of the surface. If the surface is MIP-mapped, this field contains the height of the base MIP level. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 20%;">Description</th> <th style="width: 50%;">Exists If</th> </tr> </thead> <tbody> <tr> <td>[0, 16383]</td> <td>Legal Range</td> <td>Height of surface - 1 (y/v dimension)</td> <td>(Structure[RENDER_SURFACE_STATE][Surface Type] == 'SURFTYPE_2D')</td> </tr> <tr> <td>[0, 16383]</td> <td>Legal Range</td> <td>y/v dimension</td> <td>(Structure[RENDER_SURFACE_STATE][Surface Type] == 'SURFTYPE_CUBE')</td> </tr> </tbody> </table>	Value	Name	Description	Exists If	[0, 16383]	Legal Range	Height of surface - 1 (y/v dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type] == 'SURFTYPE_2D')	[0, 16383]	Legal Range	y/v dimension	(Structure[RENDER_SURFACE_STATE][Surface Type] == 'SURFTYPE_CUBE')
Value	Name	Description	Exists If										
[0, 16383]	Legal Range	Height of surface - 1 (y/v dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type] == 'SURFTYPE_2D')										
[0, 16383]	Legal Range	y/v dimension	(Structure[RENDER_SURFACE_STATE][Surface Type] == 'SURFTYPE_CUBE')										

3DSTATE_DEPTH_BUFFER			
		Programming Notes	
		<p>The Height of the depth buffer must be the same as the Height of the</p> <ol style="list-style-type: none"> render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped) or if SURFACE_STATE_SURFTYPE is NULL. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless Stencil buffer surf_type is SURFTYPE_NULL 	
	16	Reserved	
		Format:	MBZ
	15	Reserved	
		Format:	MBZ
	14:1	Width	
		Format:	U14-1
		<p>This field specifies the width of the surface. If the surface is MIP-mapped, this field specifies the width of the base MIP level. The width is specified in units of pixels.</p>	
		Value	Name
		Description	Exists If
		[0,16383]	Legal Range
		Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]== 'SURFTYPE_2D')
		[0,16383]	Legal Range
		Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]== 'SURFTYPE_CUBE')
		Programming Notes	
		<p>The Width specified by this field must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). For cube maps, Width must be set equal to Height.</p> <p>The Width of the depth buffer must be the same as the</p> <ol style="list-style-type: none"> Width of the render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_1D or SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped) or if SURFACE_STATE_SURFTYPE is NULL. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless Stencil buffer surf_type is SURFTYPE_NULL 	
	0	Reserved	
		Format:	MBZ
5	31	Reserved	
		Format:	MBZ
	30:20	Depth	
		Format:	U11-1
		<p>This field specifies the total number of levels for a volume texture or the number of array elements allowed to be accessed starting at the Minimum Array Element for arrayed surfaces. If the volume texture is MIP-mapped, this field specifies the depth of the base MIP level.</p>	
		Value	Name
		Description	Exists If

3DSTATE_DEPTH_BUFFER			
	[0,2047]	Legal Range	Number of array elements - 1 (Structure[RENDER_SURFACE_STATE][Surface Type]='SURFTYPE_2D')
	[0,0]	Legal Range	Must be zero (Structure[RENDER_SURFACE_STATE][Surface Type]='SURFTYPE_CUBE')
Programming Notes			
<p>The Depth of the depth buffer must be the same as</p> <ol style="list-style-type: none"> The Depth of the render target(s) (defined in SURFACE_STATE). Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless Stencil buffer surf_type is SURFTYPE_NULL 			
19	Reserved		
	Format:		MBZ
18:8	Minimum Array Element		
	Format:		U11
<p>For 2D Surfaces: This field indicates the minimum array element that can be accessed as part of this surface. The delivered array index is added to this field before being used to address the surface. For Other Surfaces This field is ignored</p>			
	Value	Name	Exists If
	[0,2047]	SURFTYPE_2D	(Structure[RENDER_SURFACE_STATE][Surface Type]='SURFTYPE_1D')Structure[RENDER_SURFACE_STATE][Surface Type]='SURFTYPE_2D')
Programming Notes			
<p>Minimum array element of the depth buffer must be the same as the Surface Type of the</p> <ol style="list-style-type: none"> Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL 			
7	Reserved		
	Format:		MBZ
6:0	Depth Buffer Object Control State		
	Format:	MEMORY_OBJECT_CONTROL_STATE	
Specifies the memory object control state for the depth buffer.			
6	31:30	Tiled Mode	
For Depth Buffer Surfaces: This field specifies the tiled mode. For other surfaces: This field is ignored.			
	Value	Name	Description
	0h	NONE	No tiled modes (TileYf, TileYs). Use Standard TileY
	1h	TILEYF	4KB tiled mode

3DSTATE_DEPTH_BUFFER

		2h	TILEYS	64KB tiled mode		
		3h	Reserved			
		Programming Notes				
		If Tile Mode is not set to TILEMODE_YMAJOR, this field must be set to TRMODE_NONE.				
29:26	Mip Tail Start LOD					
		Format:	U4			
	<p>For Sampling Engine, Render Target, and Typed Surfaces: This field indicates which LOD is the first one in the MIP tail if Tiled Mode is not TRMODE_NONE. The MIP tail has a different layout than the rest of the surface. Refer to the <i>Memory Data Formats</i> section for more details.</p> <p>For other surfaces: This field is ignored.</p>					
	Programming Notes					
	<p>This field must be zero if the Surface Format is MONO8.</p> <p>This field is ignored if Tiled Mode is TRMODE_NONE unless Surface Type is SURFTYPE_1D.</p> <p>If Tiled Mode is not TRMODE_NONE, this field must be set to ensure that mips within the mip tail do not overlap given the storage algorithms given in the Memory Data Formats section. The following table indicates the maximum size of the mip that is set to be the Mip Tail Start LOD for various cases:</p>					
		Tiling Mode	Slot Size in Bytes	8-bit Size	16-bit Size	32-bit Size
		2D TileYs 1x	32KB	(128, 256)	(128, 128)	(64, 128)
		2D TileYf 1x	2KB	(32, 64)	(32, 32)	(16, 32)
25:6	Reserved					
		Format:	MBZ			
5:0	Reserved					
		Format:	MBZ			
7	31:21	Render Target View Extent				
		Format:	U11-1			
		Value	Name	Description	Exists If	
		[0,2047]	Legal Range	Number of array elements- 1	(Structure[RENDER_SURFACE_STATE][Surface Type]='SURFTYPE_2D')	
		[0,0]	Legal Range	Must be zero	(Structure[RENDER_SURFACE_STATE][Surface Type]='SURFTYPE_CUBE')	
		Programming Notes				
		<p>Render target View Extent of the depth buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL</p> <p>2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil</p>				

3DSTATE_DEPTH_BUFFER		
	buffer surf_type are SURFTYPE_NULL	
20	Reserved	
	Format:	MBZ
19:16	Surf LOD	
	Value	Name
	[0-14]	
	Programming Notes	
	Surf LOD of the depth buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL	
15	Reserved	
	Format:	MBZ
14:0	Surface QPitch	
	Format:	U17[16:2]
	Format: QPitch[16:2] The interpretation of this field is dependent on Surface Type as follows:	
	<ul style="list-style-type: none"> • SURFTYPE_2D/CUBE: distance in <i>rows</i> between array slices. 	
	Other surface types: field is ignored	
	Value	Name
	[1h,7FFFh]	in multiples of 4 (low 2 bits missing)
	Programming Notes	
	For 2D Surfaces: This field must be set to the same value as the Depth field. For Other Surfaces, This field is ignored. Refer to Alignment Unit Size for alignment sizes based on MSAA and Depth-format. Software must ensure that this field is set to a value sufficiently large that array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored. TYS/TYF QPitch is valid only for 2D array surfaces and represents the tile-padded total number of texels(lines) in a single array slice. Height of each LOD: HL = AlignToTileHeight(MSAA_height_factor * (height »L) > 0?height»L : 1), where AlignToTileHeight(x) is (ceiling((x) / (1 « Cv)) *(1 « Cv)) Height of all LODs is a sum: H = H0+H1+..Hn, N is number of mip levels.	

3DSTATE_DEPTH_BUFFER

		If surface has MIP tail, equation stops at H_n where $n = \text{MipTailStartLOD}$. MipTail is single tile. QPitch is multiple of tile height ($1 \ll C_v$) and should be equal or greater H computed above.
--	--	--

3DSTATE_DRAWING_RECTANGLE

3DSTATE_DRAWING_RECTANGLE			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_DRAWING_RECTANGLE command is used to set the 3D drawing rectangle and related state.			
Restriction			
When executed in POCS command stream, this programs the drawing rectangle for the SFR stage of the POCS pipeline.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	1h 3DSTATE_NONPIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		00h 3DSTATE_DRAWING_RECTANGLE	
Format:		OpCode	
15:14	Core Mode Select		
	Format:	U2	
	Specifies which core this command will be considered valid and update based on the state in this command.		
	Value	Name	Description
	0h	Legacy	Both cores are enabled and will update the state.
	1h	Core 0 Enabled	State will be updated in Core 0 only
	2h	Core 1 Enabled	State will be updated in Core 1 only
3h	Reserved		
13:8	Reserved		
	Format:	MBZ	
	DWord Length		
7:0	Default Value:	2h Excludes DWord (0,1)	
	Format:	=n	
	1	31:16	Clipped Drawing Rectangle Y Min

3DSTATE_DRAWING_RECTANGLE

		Format:	U16
		Specifies Ymin value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with Y coordinates less than Ymin will be clipped out.	
		Value	Name
		[0,16383]	Device ignores bits 31:30
		Programming Notes	
		This value can be larger than Clipped Drawing Rectangle Y Max. If Ymin>Ymax, the clipped drawing rectangle is null, all polygons are discarded. If Ymin==Ymax, the clipped drawing rectangle is 1 pixel wide in the Y direction.	
	15:0	Clipped Drawing Rectangle X Min	
		Format:	U16
		Specifies Xmin value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with X coordinates less than Xmin will be clipped out.	
		Value	Name
		[0,16383]	Device ignores bits 15:14
		Programming Notes	
		This value can be larger than Clipped Drawing Rectangle X Max. If Xmin>Xmax, the clipped drawing rectangle is null, all polygons are discarded. If Xmin==Xmax, the clipped drawing rectangle is 1 pixel wide in the X direction.	
2	31:16	Clipped Drawing Rectangle Y Max	
		Format:	U16
		Specifies Ymax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Ymax will be clipped out.	
		Value	Name
		[0,16383]	Device ignores bits 31:30
		Programming Notes	
		This value can be less than Clipped Drawing Rectangle Y Min. If Ymax<Ymin, the clipped drawing rectangle is null, all polygons are discarded. If Ymin==Ymax, the clipped drawing rectangle is 1 pixel wide in the Y direction.	
	15:0	Clipped Drawing Rectangle X Max	
		Format:	U16
		Specifies Xmax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Xmax will be clipped out.	
		Value	Name
		[0,16383]	Device ignores bits 15:14
		Programming Notes	

3DSTATE_DRAWING_RECTANGLE			
	<p>This value can be less than Clipped Drawing Rectangle X Min. If $X_{max} < X_{min}$, the clipped drawing rectangle is null, all polygons are discarded. If $X_{min} == X_{max}$, the clipped drawing rectangle is 1 pixel wide in the X direction.</p>		
3	<p>31:16 Drawing Rectangle Origin Y</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">S15</td> </tr> </table> <p>Range: [-16384,16383] (Bit 31 should be a sign extension)</p> <p>Specifies Y origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.</p>	Format:	S15
	Format:	S15	
<p>15:0 Drawing Rectangle Origin X</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">S15</td> </tr> </table> <p>Range: [-16384,16383] (Bit 15 should be a sign extension)</p> <p>Specifies X origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.</p>	Format:	S15	
Format:	S15		

3DSTATE_DS

3DSTATE_DS		
Source:	RenderCS	
Length Bias:	2	
The state used by DS is defined with this inline state packet		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED
23:16	3D Command Sub Opcode	
	Default Value: 1Dh 3DSTATE_DS	
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 9h Excludes DWord (0,1)	
	Format: =n	
1..10	319:0	DS State Body
	Format:	3DSTATE_DS_BODY

3DSTATE_GS

3DSTATE_GS		
Source:	RenderCS	
Length Bias:	2	
Controls the GS stage hardware.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
		Format: OpCode
	26:24	3D Command Opcode
Default Value: 0h 3DSTATE_PIPELINED		
Format: OpCode		
23:16	3D Command Sub Opcode	
	Default Value: 11h 3DSTATE_GS	
	Format: OpCode	
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 8h Excludes DWord (0,1)	
	Format: =n	
1..9	287:0	GS State Body
		Format: 3DSTATE_GS_BODY

3DSTATE_HIER_DEPTH_BUFFER

3DSTATE_HIER_DEPTH_BUFFER		
Source:	RenderCS	
Length Bias:	2	
<p>This command sets the surface state of the hierarchical depth buffer, delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).</p> <p>WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
		Format: OpCode
	26:24	3D Command Opcode
Default Value: 0h 3DSTATE_PIPELINED		
Format: OpCode		
23:16	3D Command Sub Opcode	
	Default Value: 07h 3DSTATE_HIER_DEPTH_BUFFER	
	Format: OpCode	
15:8	Reserved	
	Format: MBZ	
7:0	Dword Length	
	Default Value: 3h Excludes Dword (0,1)	
	Format: =n	
1..4	127:0	Hier Depth Buffer State Body
		Format: 3DSTATE_HIER_DEPTH_BUFFER_BODY

3DSTATE_HS

3DSTATE_HS			
Source:	RenderCS		
Length Bias:	2		
Controls the HS stage hardware.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	1Bh 3DSTATE_HS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	7 Excludes DWord (0,1)	
	Format:	=n	
1..8	255:0	HS State Body	
		Format:	3DSTATE_HS_BODY

3DSTATE_INDEX_BUFFER

3DSTATE_INDEX_BUFFER		
Source:	RenderCS	
Length Bias:	2	
<p>This command is used to specify the current IB state used by the VF function. At most one IB is defined and active at any given time. NOTES: The IB must be specified before any RANDOM 3D_PRIMITIVE commands are issued It is possible to have vertex elements source completely from generated ID values and therefore not require any Index Buffer accesses. In this case, VF function will simply ignore the Index Buffer state.</p>		
Programming Notes		
<p>When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
		Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED
Format: OpCode		
23:16	3D Command Sub Opcode	
	Default Value: 0Ah 3DSTATE_INDEX_BUFFER	
	Format: OpCode	
15	Reserved	
	Format: MBZ	
14:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 3h Excludes DWord (0,1)	
	Format: =n	
1..4	127:0	Index Buffer State Body
		Format: 3DSTATE_INDEX_BUFFER_BODY

3DSTATE_LINE_STIPPLE

3DSTATE_LINE_STIPPLE			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_LINE_STIPPLE command is used to specify state variables used in the Line Stipple function.			
Workaround			
Workaround: This command must be followed by a PIPE_CONTROL with CS Stall bit set.,			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		1h 3DSTATE_NONPIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	08h 3DSTATE_LINE_STIPPLE	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	Dword Length		
	Default Value:	1h Excludes Dword (0,1)	
	Format:	=n	
1	31	Modify Enable (Current Repeat Counter, Current Stipple Index)	
		Format:	Enable
	Modify enable for Current Repeat Counter and Current Stipple Index fields.		
	Programming Notes		
It is provided only for HW-generated commands as part of context save/restore. SW must initialize the current repeat counter, current stipple count fields if it sets this bit to enable. SW must set this bit to reset the stipple count.			
30	Reserved		
	Format:	MBZ	
29:21	Current Repeat Counter		

3DSTATE_LINE_STIPPLE									
	<table border="1"> <tr> <td>Format:</td> <td>U9</td> </tr> <tr> <td colspan="2">This field sets the HW-internal repeat counter state. SW must initialize it to 1 if the modify enable is set.</td> </tr> </table>	Format:	U9	This field sets the HW-internal repeat counter state. SW must initialize it to 1 if the modify enable is set.					
Format:	U9								
This field sets the HW-internal repeat counter state. SW must initialize it to 1 if the modify enable is set.									
20	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
19:16	<p>Current Stipple Index</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <td colspan="2">This field sets the HW-internal stipple pattern index. SW must initialize it to 0 if the modify enable is set.</td> </tr> </table>	Format:	U4	This field sets the HW-internal stipple pattern index. SW must initialize it to 0 if the modify enable is set.					
Format:	U4								
This field sets the HW-internal stipple pattern index. SW must initialize it to 0 if the modify enable is set.									
15:0	<p>Line Stipple Pattern</p> <table border="1"> <tr> <td>Format:</td> <td>Enable[16]</td> </tr> <tr> <td colspan="2">Specifies a pattern used to mask out bit specific pixels while rendering lines.</td> </tr> </table>	Format:	Enable[16]	Specifies a pattern used to mask out bit specific pixels while rendering lines.					
Format:	Enable[16]								
Specifies a pattern used to mask out bit specific pixels while rendering lines.									
2	<p>31:15 Line Stipple Inverse Repeat Count</p> <table border="1"> <tr> <td>Format:</td> <td>U1.16</td> </tr> <tr> <td colspan="2">Range: [0.00390625, 1.0]</td> </tr> <tr> <td colspan="2">Specifies the inverse (truncated) of the repeat count for the line stipple function.</td> </tr> </table>	Format:	U1.16	Range: [0.00390625, 1.0]		Specifies the inverse (truncated) of the repeat count for the line stipple function.			
	Format:	U1.16							
	Range: [0.00390625, 1.0]								
Specifies the inverse (truncated) of the repeat count for the line stipple function.									
14:9	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
8:0	<p>Line Stipple Repeat Count</p> <table border="1"> <tr> <td>Format:</td> <td>U9</td> </tr> <tr> <td colspan="2">Specifies the repeat count for the line stipple function.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> </tr> <tr> <td>[1, 256]</td> <td></td> </tr> </table>	Format:	U9	Specifies the repeat count for the line stipple function.		Value	Name	[1, 256]	
Format:	U9								
Specifies the repeat count for the line stipple function.									
Value	Name								
[1, 256]									

3DSTATE_MULTISAMPLE

3DSTATE_MULTISAMPLE		
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_MULTISAMPLE command is used to specify multisample state associated with the current render target/depth buffer/stencil buffer.		
Programming Notes		
It is illegal to render to surfaces with multiple different values of the state fields in this command.		
Restriction : When executed in the POCS command stream, this command programs the multisample state for the Raster stage of the POCS pipeline		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
Default Value: 0Dh 3DSTATE_MULTISAMPLE Format: OpCode		
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	Default Value: 0h Excludes DWord (0,1)
		Format: =n
1	31:0	Multisample State Body
		Format: 3DSTATE_MULTISAMPLE_BODY

3DSTATE_POLY_STIPPLE_OFFSET

3DSTATE_POLY_STIPPLE_OFFSET						
Source:	RenderCS					
Length Bias:	2					
The 3DSTATE_POLY_STIPPLE_OFFSET command is used to specify the origin of the repeated screen-space Polygon Stipple Pattern as an X, Y offset from the Color Buffer origin.						
Workaround						
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,						
DWord	Bit	Description				
0	31:29	Command Type				
		Default Value:	3h GFXPIPE			
		Format:	OpCode			
	28:27	Command SubType				
		Default Value:	3h GFXPIPE_3D			
		Format:	OpCode			
	26:24	3D Command Opcode				
Default Value:		1h 3DSTATE_NONPIPELINED				
Format:		OpCode				
23:16	3D Command Sub Opcode					
	Default Value:	06h 3DSTATE_POLY_STIPPLE_OFFSET				
	Format:	OpCode				
15:8	Reserved					
	Format:	MBZ				
7:0	Dword Length					
	Default Value:	0h Excludes Dword (0,1)				
	Format:	=n				
1	31:13	Reserved				
		Format:	MBZ			
	12:8	Polygon Stipple X Offset				
		Format:	U5			
		Specifies a 5 bit x address offset in the poly stipple pattern				
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,31]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,31]	
Value	Name					
[0,31]						
7:5	Reserved					
	Format:	MBZ				
4:0	Polygon Stipple Y Offset					

3DSTATE_PLY_STIPPLE_OFFSET									
	<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U5</td> </tr> <tr> <td colspan="2">Specifies a 5 bit y address offset in the poly stipple pattern</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td>[0,31]</td> <td></td> </tr> </table>	Format:	U5	Specifies a 5 bit y address offset in the poly stipple pattern		Value	Name	[0,31]	
Format:	U5								
Specifies a 5 bit y address offset in the poly stipple pattern									
Value	Name								
[0,31]									

3DSTATE_POLY_STIPPLE_PATTERN

3DSTATE_POLY_STIPPLE_PATTERN			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_POLY_STIPPLE_PATTERN command is used to specify the 32x32 Polygon Stipple Pattern used in the Polygon Stipple function of the WM unit.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		1h 3DSTATE_NONPIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	07h 3DSTATE_POLY_STIPPLE_PATTERN	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	Dword Length		
	Default Value:	1Fh Excludes Dword (0,1)	
	Format:	=n	
1..32	1023:0	Pattern Row	
		Format:	U32[32]
Specifies a pattern used by Polygon Stipple to mask out specific pixels of every 32x32 area rendered.			

3DSTATE_PRIMITIVE_REPLICATION

3DSTATE_PRIMITIVE_REPLICATION			
Source:		RenderCS, PositionCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		6Ch 3DSTATE_PRIMITIVE_REPLICATION	
Format:		OpCode	
15:11	Reserved		
	Format:	MBZ	
10:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	4h Excludes DWord (0,1)	
	Format:	=n	
1..5	159:0	Primitive Replication State Body	
		Format:	3DSTATE_PRIMITIVE_REPLICATION_BODY

3DSTATE_PS_BLEND

3DSTATE_PS_BLEND			
Source:	RenderCS		
Length Bias:	2		
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	4Dh 3DSTATE_PS_BLEND	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1	31:0	PS Blend State Body	
		Format:	3DSTATE_PS_BLEND_BODY

3DSTATE_PS_EXTRA

3DSTATE_PS_EXTRA			
Source:		RenderCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		4fh 3DSTATE_PS_EXTRA	
Format:		OpCode	
15	POSH-Enabled Render State Optimization Enable 1		
14:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1	31:0	PS Extra State Body	
		Format:	3DSTATE_PS_EXTRA_BODY

3DSTATE_PS

3DSTATE_PS		
Source:		RenderCS
Length Bias:		2
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
		Default Value: 20h 3DSTATE_PS Format: OpCode
15	Reserved	
Format: MBZ		
14:8	Reserved	
Format: MBZ		
7:0	DWord Length	
	Default Value: 0Ah Excludes DWord (0,1) Format: =n	
1..11	351:0	PS State Body
Format: 3DSTATE_PS_BODY		

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS		
Source:	PositionCS	
Length Bias:	2	
<p>The 3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS command programmed specifies a 4KB aligned base address to 128KB ($2^{16} \times 2B$) of contiguous memory surface called Free-List memory. This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context.</p> <p>POSH pipe uses Free-List memory to track the free pages (pageoffsets from PTBR Page Pool Base Address are stored) available for recording visibility data. Pages get consumed from the Free-List as and when the visibility data is recorded by POSH pipe and pages get added to the Free-List when freed up by the render engine on rendering.</p>		
Programming Notes		
The Free List Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
	Format: OpCode	
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
	Format: OpCode	
26:24	3D Command Opcode	
	Default Value: 1h 3DSTATE_NONPIPELINED	
Format: OpCode		
23:16	3D Command Sub Opcode	
	Default Value: 24h 3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS	
Format: OpCode		
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 01h Excludes DWord (1,2)	
	Format: =n	
1..2	63:12	Free List Base Address
		Format: VIRTUAL_ADDR[63:12] The Free List Base Address specifies a 4KB aligned base address for a Free-List memory.

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS		
	11:7	Reserved Format: MBZ
	6:0	Free List Memory Object Control State Format: MEMORY_OBJECT_CONTROL_STATE Specifies the MOCS state for Free-List memory accesses using PTBR_FREE_LIST_BASE_ADDRESS.

3DSTATE_PTBR_MARKER

3DSTATE_PTBR_MARKER		
Source:	RenderCS	
Length Bias:	2	
Programming Notes		
Start of Tile and End of Tile must not be set in the same command. Every Start of Tile marker programmed must have a corresponding End of Tile marker programmed.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
Default Value: 6Ah 3DSTATE_PTBR_MARKER Format: OpCode		
15:8	Reserved	
7:0	DWord Length	
1	31:0	PTBR Marker State Body
		Format: 3DSTATE_PTBR_MARKER_BODY



3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS		
Source:	PositionCS	
Length Bias:	2	
<p>The 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command specifies base address of the PTBR page pool allocated to HW for recording visibility data in POSH Tile Based Rendering Mode (PTBR). The size of the PTBR Page Pool Size is programmed through the MMIO register PTBR_PAGE_POOL_SIZE_REGISTER following this command. SW must always set Page Pool Restart to enabled when programming this command. This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context. HW context save/restores 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command with Page Pool Restart set to disabled.</p>		
Programming Notes		
The Page Pool Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
Default Value: 21h 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS Format: OpCode		
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 01h Excludes DWord (1,2) Format: =n	
1..2	63:12	Page Pool Base Address Format: VIRTUAL_ADDR[63:12] Page Pool Base Address specifies the 4KB aligned base address of the contiguous graphics memory allocated to HW for recording visibility data in POSH Tile Based Rendering Mode (PTBR).

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS				
	11	Page Pool Restart Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td>Enable</td></tr></table> <ul style="list-style-type: none"> SW must always set Page Pool Restart to enabled on programming 3DSTATE_PTBR_PAGE_POOL_ADDRESS command. SW must ensure both POSH and Render pipe are flushed and synchronized prior to programming 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command, this ensures there aren't any pages in use on the render side from the old PTBR page pool when this command is parsed by HW. 		Enable
		Enable		
	10:7	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td>MBZ</td></tr></table>		MBZ
	MBZ			
6:0	Page Pool Memory Object Control State Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 50px;"> </td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table> Specifies the MOCS state for PTBR Page Pool memory accesses using Page Pool Base Address.		MEMORY_OBJECT_CONTROL_STATE	
	MEMORY_OBJECT_CONTROL_STATE			
3	31:16	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td>MBZ</td></tr></table>		MBZ
		MBZ		
15:0	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS			
Source:	PositionCS		
Length Bias:	2		
<p>The 3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS command specifies a 4KB aligned base address for storing Render-Lists in PTBR mode. Render-List is an array consisting of starting page offset's corresponding to each of the tiles visibility data. Starting page offsets stored are offsets relative to Page Pool Base Address. The number of entries (array size) in a Render-List is equivalent to the number of tiles in a Tile Pass. Each Tile Pass has a corresponding Render-List populated by POSH pipe and each entry from the Render-List is read by Render Pipe when the corresponding tile is rendered.</p>			
Programming Notes			
<p>Both POSH pipe and Render pipe use the Render List Base Address programmed in POSH pipe, SW must ensure both POSH and Render pipe are flushed and synchronized prior to programming this command. This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context.</p>			
<p>The Render List Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		1h 3DSTATE_NONPIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	23h 3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	01h Excludes DWord (1,2)	
	Format:	=n	
1..2	63:12	Render List Base Address	
		Format: VIRTUAL_ADDR[63:12]	

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS			
	Render List Base Address specifies a 4KB aligned base address for storing Render-Lists in PTBR mode.		
11:7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
6:0	<p>Render List Memory Object Control State</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control start for using the Render-List Address.</p>	Format:	MEMORY_OBJECT_CONTROL_STATE
Format:	MEMORY_OBJECT_CONTROL_STATE		



3DSTATE_PTBR_TILE_PASS_INFO

DWord		Bit	Description												
Source:		RenderCS, PositionCS													
Length Bias:		2													
The 3DSTATE_PTBR_TILE_PASS_INFO command is used to define the required parameters for a Tile Pass in PTBR mode.															
Programming Notes															
This command must be programmed for both POSH pipe and render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context by POSH pipe and gets context save/restored as part of the render context by render pipe.															
RENDER PIPE ONLY: PIPE_CONTROL DepthStallEnable and DepthCacheFlushEnable are required before the TILE_PASS_INFO.EndofTilePass marker.															
0	31:29	Command Type	<table border="1"> <tr> <td>Default Value:</td> <td>3h GFXPIPE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	3h GFXPIPE	Format:	OpCode								
Default Value:	3h GFXPIPE														
Format:	OpCode														
	28:27	Command SubType	<table border="1"> <tr> <td>Default Value:</td> <td>3h GFXPIPE_3D</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	3h GFXPIPE_3D	Format:	OpCode								
Default Value:	3h GFXPIPE_3D														
Format:	OpCode														
	26:24	3D Command Opcode	<table border="1"> <tr> <td>Default Value:</td> <td>1h 3DSTATE_NONPIPELINED</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	1h 3DSTATE_NONPIPELINED	Format:	OpCode								
Default Value:	1h 3DSTATE_NONPIPELINED														
Format:	OpCode														
	23:16	3D Command Sub Opcode	<table border="1"> <tr> <td>Default Value:</td> <td>22h 3DSTATE_PTBR_TILE_PASS_INFO</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	22h 3DSTATE_PTBR_TILE_PASS_INFO	Format:	OpCode								
Default Value:	22h 3DSTATE_PTBR_TILE_PASS_INFO														
Format:	OpCode														
	15:8	Reserved	<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ														
	7:0	DWord Length	<table border="1"> <tr> <td>Format:</td> <td>=n</td> </tr> <tr> <td>Value</td> <td>Name</td> </tr> <tr> <td>02h</td> <td>End Of Tile Pass Marker Not Set [Default]</td> </tr> <tr> <td>0h</td> <td>End Of Tile Pass Marker Set</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">On setting the End Of;Tile Pass Marker, SW must not program the Dword2 and Dword3 of the</td> </tr> </table>	Format:	=n	Value	Name	02h	End Of Tile Pass Marker Not Set [Default]	0h	End Of Tile Pass Marker Set	Programming Notes		On setting the End Of;Tile Pass Marker, SW must not program the Dword2 and Dword3 of the	
Format:	=n														
Value	Name														
02h	End Of Tile Pass Marker Not Set [Default]														
0h	End Of Tile Pass Marker Set														
Programming Notes															
On setting the End Of;Tile Pass Marker, SW must not program the Dword2 and Dword3 of the															

3DSTATE_PTBR_TILE_PASS_INFO											
		command and must accordingly program the Dword count in the header.									
1	31	End of Tile Pass Format: _____ Enable Marker for indicating End of Tile Pass in a command sequence. This bit is always save/restored as 0 in HW CONTEXT.									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>[Default]</td> <td>Not an indicator of End of Tile Pass in a command sequence.</td> </tr> <tr> <td>1</td> <td></td> <td>Indicates Start of Tile Pass in a command sequence.</td> </tr> </tbody> </table>	Value	Name	Description	0	[Default]	Not an indicator of End of Tile Pass in a command sequence.	1		Indicates Start of Tile Pass in a command sequence.
		Value	Name	Description							
0	[Default]	Not an indicator of End of Tile Pass in a command sequence.									
1		Indicates Start of Tile Pass in a command sequence.									
<p style="text-align: center;">Programming Notes</p> <ul style="list-style-type: none"> • A Tile Pass must begin with a Start of Tile Pass marker and end with End of Tile Pass marker. • Start of Tile Pass marker and End of Tile Pass marker must not be set in the same command • On setting the End of Tile Pass Marker, SW must not program the Dword2 and Dword3 of the command and must accordingly program the Dword count in the header. 											
	30	Start of Tile Pass Format: _____ Enable Marker for indicating Start of Tile Pass in a command sequence.									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>[Default]</td> <td>Not an indicator of Start of Tile Pass in a command sequence.</td> </tr> <tr> <td>1</td> <td></td> <td>Indicates Start of Tile Pass in a command sequence.</td> </tr> </tbody> </table>	Value	Name	Description	0	[Default]	Not an indicator of Start of Tile Pass in a command sequence.	1		Indicates Start of Tile Pass in a command sequence.
		Value	Name	Description							
0	[Default]	Not an indicator of Start of Tile Pass in a command sequence.									
1		Indicates Start of Tile Pass in a command sequence.									
<p style="text-align: center;">Programming Notes</p> <ul style="list-style-type: none"> • A Tile Pass must begin with a Start of Tile Pass marker and end with End of Tile Pass marker. • Start of Tile Pass marker and End of Tile Pass marker must not be set in the same command 											
	29	Tile Pass Flag Status Format: _____ Enable This bit is used by hardware to context save/restore the Tile Pass Flag status.									
		<ul style="list-style-type: none"> • When set indicates context got switched out is in middle of a Tile Pass, • when reset indicates context got switched out outside the Tile Pass. 									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>[Default]</td> <td>Context got switched out outside the Tile Pass.</td> </tr> <tr> <td>1</td> <td></td> <td>Context got switched out in middle of a Tile Pass.</td> </tr> </tbody> </table>	Value	Name	Description	0	[Default]	Context got switched out outside the Tile Pass.	1		Context got switched out in middle of a Tile Pass.
Value	Name	Description									
0	[Default]	Context got switched out outside the Tile Pass.									
1		Context got switched out in middle of a Tile Pass.									

3DSTATE_PTBR_TILE_PASS_INFO														
		Programming Notes												
		SW must never set this bit.												
	28:10	Reserved												
		Format: MBZ												
	9:0	Tile Count												
		Format: U10-1												
		Description												
		<p>Tile Count indicates the number of tiles the render region is divided in to, a maximum of 128 tiles is supported by POSH pipe and a maximum of 1024 tiles by Render pipe.</p> <p>A tile count of zero is an special indication stating the whole render region is considered as one single tile of the dimensions of the drawing rectangle. A tile count of 1 indicates the whole render region is divided in to two tiles and tile count of 127 indicates the render region is divided in to 128 tiles.</p> <p>For a Tile Count greater then '0', Tile Rect Array Pointer provides the details for each of the tiles dimensions.</p>												
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Default [Default]</td> <td>The whole render region is considered as one single tile of the dimensions of the drawing rectangle.</td> </tr> <tr> <td>7Fh</td> <td>POSH Max Value</td> <td>The render region is divided in to 128 tiles.</td> </tr> <tr> <td>3FFh</td> <td>Render Max Value</td> <td>The render region is divided in to 1024 tiles.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Default [Default]	The whole render region is considered as one single tile of the dimensions of the drawing rectangle.	7Fh	POSH Max Value	The render region is divided in to 128 tiles.	3FFh	Render Max Value	The render region is divided in to 1024 tiles.
Value	Name	Description												
0h	Default [Default]	The whole render region is considered as one single tile of the dimensions of the drawing rectangle.												
7Fh	POSH Max Value	The render region is divided in to 128 tiles.												
3FFh	Render Max Value	The render region is divided in to 1024 tiles.												
		Programming Notes												
		"Tile Count" value programmed must be same in the 3DSTATE_PTBR_TILE_PASS_INFO command programmed for "Start of Tile Pass" and for "End of Tile Pass".												
2	31:6	Tile Rect Array Pointer												
		Format: DynamicStateOffset[31:6]TILE_RECT												
		<p>Specifies the 64-byte aligned offset pointing to the element zero of an array, each element is a TILE_RECT state. This offset is relative to the Dynamic State Base Address. Each TILE_RECT state is two dwords in size comprising {Y Max, X Max, Y Min, X Min} of the tile. Number of TILE_RECT states in the array is equivalent to the tile count. Example: For a Tile count of 63, 64 TILE_RECT states are programmed in consecutive memory locations constituting the array, i.e 64 consecutive qwords (qword-64bits) of data in the memory, with starting qword corresponding to the TILE_RECT state of tile0 and the last qword corresponding to tile-63.</p>												
	5:0	Reserved												
		Format: MBZ												
3	31:6	Render List Pointer												

3DSTATE_PTBR_TILE_PASS_INFO

		Format:	RenderListOffset[31:6]
		<p>Specifies the 64byte offset pointing to Render-List of the current Tile Pass. This offset is relative to PTBR_RENDER_LIST_BASE_ADDRESS. Render-List is an array consisting of starting page offset's corresponding to each of the tiles visibility data. Starting page offsets stored are 16bit offsets relative to PTBR_PAGE_POOL_BASE_ADDRESS. The number of entries (array size) in a Render-List is equivalent to the tile count of the current Tile Pass.</p>	
5:2	Reserved	Format:	MBZ
1	Reserved	Format:	MBZ
0	Initialize Render List	Format:	Enable
		<p>On parsing this command with Initialize Render List set, POSH pipe allocates a page for each of the tile from the Free-List to which respective visibility data will be outputted. The page offsets corresponding to these pages are written to the Render-List. POSH pipe doesn't parse the next command in sequence until the Render-List is populated. Each entry from the Render-List is read by Render Pipe when the corresponding tile is rendered. Example: For a Tile count of 63, POSH pipe will write to 64 consecutive Words (2bytes) to the Render-List, where in word-0 belongs to Tile-0, word1 belongs to Tile-1 word-63 belongs to Tile-63.</p> <p>This bit is ignored by RenderCS on executing 3DSTATE_PTBR_TILE_PASS_INFO command.</p>	
		Programming Notes	
		<p>SW must ensure Render-List is populated by POSH pipe before it gets read by the Render pipe. This can be achieved by using semaphores in render pipe command sequence to wait for POSH pipe to complete Render-List update.</p> <ul style="list-style-type: none"> • SW must always set this bit to '1'. • SW must Initialize Render List only once in a Tile Pass. • SW must ensure to program the below listed commands prior to programming 3DSTATE_PTBR_TILE_PASS_INFO with "Initialize Render List bit set. <ul style="list-style-type: none"> • 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS • 3DSTATE_PTBR_FREE_LIST • 3DSTATE_PTBR_RENDER_LIST <p>This bit is always save/restored as 0 in HW CONTEXT.</p>	

3DSTATE_PTBR_TILE_SELECT

3DSTATE_PTBR_TILE_SELECT		
Source:	RenderCS	
Length Bias:	2	
<p>The 3DSTATE_PTBR_TILE_SELECT command is programmed in render engine to define the required parameters for a tile to be rendered in POSH Tile Base Rendering mode (PTBR). This command must be programmed only for render pipe and must not be programmed for POSH pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored by render engine as part of the render context.</p>		
Programming Notes		
<p>SW must ensure Render-List is populated by POSH pipe before it gets read by the Render pipe. This can be achieved by using semaphores in render pipe command sequence to wait for POSH pipe to complete Render-List update prior to programming of 3DSTATE_PTBR_TILE_SELECT command.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
Default Value: 6Bh 3DSTATE_PTBR_TILE_SELECT Format: OpCode		
15:8	Reserved	
7:0	DWord Length	
	Format: =n	
1	31:0	PTBR Tile Select State Body
		Format: 3DSTATE_PTBR_TILE_SELECT_BODY

3DSTATE_PUSH_CONSTANT_ALLOC_DS

3DSTATE_PUSH_CONSTANT_ALLOC_DS		
Source:	RenderCS	
Length Bias:	2	
This command sets up the URB configuration for DS Push Constant Buffer.		
Programming Notes		
Programming Restriction: <ul style="list-style-type: none"> The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size. The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See Push Constant URB Allocation section for more details. The Domain Shader Push Constants state must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_DS. 		
When gather at set shader is disabled, Push Constant command is committed when 3DPRIMITIVE command is parsed. When gather at set shader is enabled, commit point on the Push Constant command is a 3DSTATE_BINDING_TABLE_POINTER command.		
The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.		
DWord	Bit	Description
0 Programming Notes: Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,	31:29	Command Type
		Default Value:
	Format:	OpCode
	28:27	Command SubType
		Default Value:
	Format:	OpCode
	26:24	3D Command Opcode
		Default Value:
	Format:	OpCode
	23:16	3D Command Sub Opcode
Default Value:		14h 3DSTATE_PUSH_CONSTANT_ALLOC_DS
Format:	OpCode	
15:8	Reserved	
	Format:	MBZ
7:0	DWord Length	

3DSTATE_PUSH_CONSTANT_ALLOC_DS					
		Default Value:	0h Excludes DWord (0,1)		
		Format:	=n		
1	31:21	Reserved			
		Format:	MBZ		
	20:16	Constant Buffer Offset			
		Format:	U5		
		Specifies the offset of the DS constant buffer into the URB.			
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,31]</td> <td>(0KB - 31KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,31]
	Value	Name			
	[0,31]	(0KB - 31KB) Increments of 2KB			
	15:6	Reserved			
		Format:	MBZ		
5:0	Constant Buffer Size				
	Format:	U6			
	Specifies the size of the DS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for DS.				
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,32]</td> <td>(0KB - 32KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,32]	(0KB - 32KB) Increments of 2KB
Value	Name				
[0,32]	(0KB - 32KB) Increments of 2KB				

3DSTATE_PUSH_CONSTANT_ALLOC_GS

3DSTATE_PUSH_CONSTANT_ALLOC_GS			
Source:	RenderCS		
Length Bias:	2		
This command sets up the URB configuration for GS Push Constant Buffer.			
Programming Notes			
<ul style="list-style-type: none"> The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size. The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See Push Constant URB Allocation section for more details. The GeometryShader Push Constantsstate must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_GS. 			
<p>When gather at set shader is disabled, Push Constantcommand is committed when 3DPRIMITIVE command is parsed.</p> <p>When gather at set shader is enabled, commit point on the Push Constantcommand is a 3DSTATE_BINDING_TABLE_POINTER command.</p>			
The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.			
Workaround			
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	1h 3DSTATE_NONPIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	15h 3DSTATE_PUSH_CONSTANT_ALLOC_GS
		Format:	OpCode
	15:8	Reserved	

3DSTATE_PUSH_CONSTANT_ALLOC_GS								
		Format: MBZ						
	7:0	DWord Length Format: =n Total Length - 2 <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>3DSTATE_PUSH_CONSTANT_ALLOC_GS [Default]</td> <td>Excludes DWord (0,1)</td> </tr> </tbody> </table>	Value	Name	Description	0h	3DSTATE_PUSH_CONSTANT_ALLOC_GS [Default]	Excludes DWord (0,1)
Value	Name	Description						
0h	3DSTATE_PUSH_CONSTANT_ALLOC_GS [Default]	Excludes DWord (0,1)						
1	31:21	Reserved Format: MBZ						
	20:16	Constant Buffer Offset Format: U5 Specifies the offset of the GS constant buffer into the URB. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,31]</td> <td>(0KB - 31KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,31]	(0KB - 31KB) Increments of 2KB		
	Value	Name						
	[0,31]	(0KB - 31KB) Increments of 2KB						
	15:6	Reserved Format: MBZ						
5:0	Constant Buffer Size Format: U6 Specifies the size of the GS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for GS. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,32]</td> <td>(0KB - 32KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,32]	(0KB - 32KB) Increments of 2KB			
Value	Name							
[0,32]	(0KB - 32KB) Increments of 2KB							

3DSTATE_PUSH_CONSTANT_ALLOC_HS

3DSTATE_PUSH_CONSTANT_ALLOC_HS		
Source:	RenderCS	
Length Bias:	2	
This command sets up the URB configuration for HS Push Constant Buffer.		
Programming Notes		
Programming Restriction: <ul style="list-style-type: none"> The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size. The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See Push Constant URB Allocation section for more details. The Hull Shader Push Constants state must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_HS. 		
When gather at set shader is disabled, Push Constant command is committed when 3DPRIMITIVE command is parsed. When gather at set shader is enabled, commit point on the Push Constant command is a 3DSTATE_BINDING_TABLE_POINTER command.		
The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.		
Workaround		
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set,		
DWord	Bit	Description
0	31:29	Command Type
		Default Value:
	Format:	OpCode
	28:27	Command SubType
		Default Value:
	Format:	OpCode
	26:24	3D Command Opcode
		Default Value:
	Format:	OpCode
	23:16	3D Command Sub Opcode
		Default Value:
	Format:	OpCode

3DSTATE_PUSH_CONSTANT_ALLOC_HS						
	15:8	Reserved Format: _____ MBZ				
	7:0	DWord Length Default Value: _____ 0h Excludes DWord (0,1) Format: _____ =n				
1	31:21	Reserved Format: _____ MBZ				
	20:16	Constant Buffer Offset Format: _____ U5 Specifies the offset of the HS constant buffer into the URB. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,31]</td> <td>(0KB - 31KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,31]	(0KB - 31KB) Increments of 2KB
	Value	Name				
	[0,31]	(0KB - 31KB) Increments of 2KB				
	15:6	Reserved Format: _____ MBZ				
5:0	Constant Buffer Size Format: _____ U6 Specifies the size of the HS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for HS. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,32]</td> <td>(0KB - 32KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,32]	(0KB - 32KB) Increments of 2KB	
Value	Name					
[0,32]	(0KB - 32KB) Increments of 2KB					

3DSTATE_PUSH_CONSTANT_ALLOC_PS

3DSTATE_PUSH_CONSTANT_ALLOC_PS		
Source:	RenderCS	
Length Bias:	2	
This command sets up the URB configuration for PS Push Constant Buffer.		
Programming Notes		
Restriction:		
<ul style="list-style-type: none"> The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size. The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See Push Constant URB Allocation section for more details. The PixelShader Push Constants state must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_PS. 		
When gather at set shader is disabled, Push Constant command is committed when 3DPRIMITIVE command is parsed.		
When gather at set shader is enabled, commit point on the Push Constant command is a 3DSTATE_BINDING_TABLE_POINTER command.		
The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.		
Workaround		
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set,		
DWord	Bit	Description
0	31:29	Command Type
		Default Value:
	Format:	OpCode
	28:27	Command SubType
		Default Value:
	Format:	OpCode
	26:24	3D Command Opcode
		Default Value:
	Format:	OpCode
	23:16	3D Command Sub Opcode
		Default Value:
	Format:	OpCode

3DSTATE_PUSH_CONSTANT_ALLOC_PS						
	15:8	Reserved Format: _____ MBZ				
	7:0	Dword Length Default Value: _____ 0h Excludes Dword (0,1) Format: _____ =n				
1	31:21	Reserved Format: _____ MBZ				
	20:16	Constant Buffer Offset Format: _____ U5 Specifies the offset of the PS constant buffer into the URB.				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,31]</td> <td>(0KB - 31KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,31]	(0KB - 31KB) Increments of 2KB
		Value	Name			
	[0,31]	(0KB - 31KB) Increments of 2KB				
15:6	Reserved Format: _____ MBZ					
5:0	Constant Buffer Size Format: _____ U6 Specifies the size of the PS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for PS.					
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,32]</td> <td>(0KB - 32KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,32]	(0KB - 32KB) Increments of 2KB	
	Value	Name				
[0,32]	(0KB - 32KB) Increments of 2KB					

3DSTATE_PUSH_CONSTANT_ALLOC_VS

3DSTATE_PUSH_CONSTANT_ALLOC_VS		
Source:	RenderCS	
Length Bias:	2	
This command sets up the URB configuration for VS Push Constant Buffer.		
Programming Notes		
<p>Programming Restriction:</p> <ul style="list-style-type: none"> The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size. The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See Push Constant URB Allocation section for more details. The VertexShader Push Constants state must be committed prior to shaders generating thread payloads after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_VS. 		
When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline. The offset & size of the VSR allocation is relative to the 32KB of URB reserved for POCS pipe Push Constant Buffers, vs. the 32KB used by the RCS pipe.		
When gather at set shader is disabled, programmed constants are committed when 3DPRIMITIVE command is parsed.		
When gather at set shader is enabled, commit point of the constants programmed area 3DSTATE_BINDING_TABLE_POINTER command.		
The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
	Format: OpCode	
	28:27	Command SubType
Default Value: 3h GFXPIPE_3D		
Format: OpCode		
26:24	3D Command Opcode	
	Default Value: 1h 3DSTATE_NONPIPELINED	
Format: OpCode		
23:16	3D Command Sub Opcode	
	Default Value: 12h 3DSTATE_PUSH_CONSTANT_ALLOC_VS	
Format: OpCode		

3DSTATE_PUSH_CONSTANT_ALLOC_VS						
	15:8	Reserved				
	Format: MBZ					
	7:0	DWord Length				
	Default Value: 0h Excludes DWord (0,1)					
		Format: =n				
1	31:21	Reserved				
	Format: MBZ					
	20:16	Constant Buffer Offset				
	Format: U5					
	Specifies the offset of the VS constant buffer into the URB.					
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,31]</td> <td>(0KB - 31KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,31]	(0KB - 31KB) Increments of 2KB
Value	Name					
[0,31]	(0KB - 31KB) Increments of 2KB					
		Programming Notes				
		When executed from the POCS pipe, the offset is relative to the VSR_PUSH_CONSTANT_BASE (MMIO offset e518) region reserved for POCS pipe Push Constants.				
	15:6	Reserved				
	Format: MBZ					
	5:0	Constant Buffer Size				
	Format: U6					
		Specifies the size of the VS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for VS.				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,32]</td> <td>(0KB - 32KB) Increments of 2KB</td> </tr> </tbody> </table>	Value	Name	[0,32]	(0KB - 32KB) Increments of 2KB
Value	Name					
[0,32]	(0KB - 32KB) Increments of 2KB					

3DSTATE_RASTER

3DSTATE_RASTER			
Source:	RenderCS		
Length Bias:	2		
Restriction			
When executed in the POCS command stream, this command programs the raster state for CLR and SFR stages of the POCS pipeline			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		50h 3DSTATE_RASTER	
Format:		OpCode	
15:14	Reserved		
	Format:	MBZ	
13	Raster State Modify Disable		
	Format:	Disable	
When this bit is set, the following fields will be ignored: <ul style="list-style-type: none"> • SubPixel Aligned Quad Rasterization Enable • Line/Point Conservative Rasterization Enable • Conservative Rasterization Enable • API Mode • Forced Sample Count • Force Multisampling • Smooth Point Enable • DX Multisample Rasterization Enable • DX Multisample Rasterization Mode • Global Depth Offset Enable Solid • Global Depth Offset Enable Wireframe 			

3DSTATE_RASTER					
	<ul style="list-style-type: none"> • Global Depth Offset Enable Point • Antialiasing Enable • Scissor Rectangle Enable 				
12	<p>Front Winding Modify Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Front Winding 	Format:	Disable		
Format:	Disable				
11	<p>Cull Mode Modify Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Cull Mode 	Format:	Disable		
Format:	Disable				
10	<p>Fill Mode Modify Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Front Face Fill Mode • Back Face Fill Mode 	Format:	Disable		
Format:	Disable				
9	<p>Viewport Z Clip Test Enable Modify Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Viewport Z Far Clip Test Enable • Viewport Z Near Clip Test Enable 	Format:	Disable		
Format:	Disable				
8	<p>Global Depth Offset Modify Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Global Depth Offset Constant • Global Depth Offset Scale • Global Depth Offset Clamp 	Format:	Disable		
Format:	Disable				
7:0	<p>DWord Length</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>03h Excludes DWord (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	03h Excludes DWord (0,1)	Format:	=n
Default Value:	03h Excludes DWord (0,1)				
Format:	=n				

3DSTATE_RASTER		
		Total Length - 2
1..4	127:0	Raster State Body Format: 3DSTATE_RASTER_BODY



3DSTATE_SAMPLE_MASK

3DSTATE_SAMPLE_MASK		
Source:	RenderCS	
Length Bias:	2	
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED
23:16	3D Command Sub Opcode	
	Default Value: 18h 3DSTATE_SAMPLE_MASK	
15:8	Reserved	
7:0	Format: MBZ	
	Dword Length	
	Default Value: 0h Excludes Dword (0,1)	
1	31:0	Sample Mask State Body
		Format: 3DSTATE_SAMPLE_MASK_BODY

3DSTATE_SAMPLE_PATTERN

3DSTATE_SAMPLE_PATTERN		
Source:	RenderCS	
Length Bias:	2	
The 3DSTATE_SAMPLE_PATTERN command is used to specify the sample offsets for all multisample sample modes. The set of offset used will be selected based on the multisample mode. This is non-pipelined state.		
Programming Notes		
When programming the sample offsets (for NUMSAMPLES_4 or _8 and MSRASTMODE_xxx_PATTERN), the order of the samples 0 to 3 (or 7 for 8X, or 15 for 16X) must have monotonically increasing distance from the pixel center. This is required to get the correct centroid computation in the device.		
Restriction : When executed in the POCS command stream, this command programs the multisample pattern state for the CLR and SFR stage of the POCS pipeline		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
	26:24	3D Command Opcode
		Default Value: 1h 3DSTATE_NONPIPELINED
23:16	3D Command Sub Opcode	
	Default Value: 1Ch 3DSTATE_SAMPLE_PATTERN	
15:8	Reserved	
7:0	Format: MBZ	
	DWord Length	
	Default Value: 7 Excludes Dword (0,1)	
1	31:28	16x Sample3 X Offset
		Format: U0.4
		Subpixel X offset of Sample 3 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	16x Sample3 Y Offset

3DSTATE_SAMPLE_PATTERN										
		<table border="1"> <tr> <td>Format:</td> <td>U0.4</td> </tr> <tr> <td colspan="2">Subpixel Y offset of Sample 3 relative to the UL pixel origin for 16x mode.</td> </tr> <tr> <td colspan="2">Range: [0,0.9375]</td> </tr> </table>	Format:	U0.4	Subpixel Y offset of Sample 3 relative to the UL pixel origin for 16x mode.		Range: [0,0.9375]			
Format:	U0.4									
Subpixel Y offset of Sample 3 relative to the UL pixel origin for 16x mode.										
Range: [0,0.9375]										
	23:20	<table border="1"> <tr> <td colspan="2">16x Sample2 X Offset</td> </tr> <tr> <td>Format:</td> <td>U0.4</td> </tr> <tr> <td colspan="2">Subpixel X offset of Sample 2 relative to the UL pixel origin for 16x mode.</td> </tr> <tr> <td colspan="2">Range: [0,0.9375]</td> </tr> </table>	16x Sample2 X Offset		Format:	U0.4	Subpixel X offset of Sample 2 relative to the UL pixel origin for 16x mode.		Range: [0,0.9375]	
16x Sample2 X Offset										
Format:	U0.4									
Subpixel X offset of Sample 2 relative to the UL pixel origin for 16x mode.										
Range: [0,0.9375]										
	19:16	<table border="1"> <tr> <td colspan="2">16x Sample2 Y Offset</td> </tr> <tr> <td>Format:</td> <td>U0.4</td> </tr> <tr> <td colspan="2">Subpixel Y offset of Sample 2 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_4, _8 or _16.</td> </tr> <tr> <td colspan="2">Range: [0,0.9375]</td> </tr> </table>	16x Sample2 Y Offset		Format:	U0.4	Subpixel Y offset of Sample 2 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_4, _8 or _16.		Range: [0,0.9375]	
16x Sample2 Y Offset										
Format:	U0.4									
Subpixel Y offset of Sample 2 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_4, _8 or _16.										
Range: [0,0.9375]										
	15:12	<table border="1"> <tr> <td colspan="2">16x Sample1 X Offset</td> </tr> <tr> <td>Format:</td> <td>U0.4</td> </tr> <tr> <td colspan="2">Subpixel X offset of Sample 1 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_2, _4, _8 or _16.</td> </tr> <tr> <td colspan="2">Range: [0,0.9375]</td> </tr> </table>	16x Sample1 X Offset		Format:	U0.4	Subpixel X offset of Sample 1 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_2, _4, _8 or _16.		Range: [0,0.9375]	
16x Sample1 X Offset										
Format:	U0.4									
Subpixel X offset of Sample 1 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_2, _4, _8 or _16.										
Range: [0,0.9375]										
	11:8	<table border="1"> <tr> <td colspan="2">16x Sample1 Y Offset</td> </tr> <tr> <td>Format:</td> <td>U0.4</td> </tr> <tr> <td colspan="2">Subpixel Y offset of Sample 1 relative to the UL pixel origin for 16x mode.</td> </tr> <tr> <td colspan="2">Range: [0,0.9375]</td> </tr> </table>	16x Sample1 Y Offset		Format:	U0.4	Subpixel Y offset of Sample 1 relative to the UL pixel origin for 16x mode.		Range: [0,0.9375]	
16x Sample1 Y Offset										
Format:	U0.4									
Subpixel Y offset of Sample 1 relative to the UL pixel origin for 16x mode.										
Range: [0,0.9375]										
	7:4	<table border="1"> <tr> <td colspan="2">16x Sample0 X Offset</td> </tr> <tr> <td>Format:</td> <td>U0.4</td> </tr> <tr> <td colspan="2">Subpixel X offset of Sample 0 relative to the UL pixel origin for 16x mode.</td> </tr> <tr> <td colspan="2">Range: [0,0.9375]</td> </tr> </table>	16x Sample0 X Offset		Format:	U0.4	Subpixel X offset of Sample 0 relative to the UL pixel origin for 16x mode.		Range: [0,0.9375]	
16x Sample0 X Offset										
Format:	U0.4									
Subpixel X offset of Sample 0 relative to the UL pixel origin for 16x mode.										
Range: [0,0.9375]										
	3:0	<table border="1"> <tr> <td colspan="2">16x Sample0 Y Offset</td> </tr> <tr> <td>Format:</td> <td>U0.4</td> </tr> <tr> <td colspan="2">Subpixel Y offset of Sample 0 relative to the UL pixel origin for 16x mode.</td> </tr> <tr> <td colspan="2">Range: [0,0.9375]</td> </tr> </table>	16x Sample0 Y Offset		Format:	U0.4	Subpixel Y offset of Sample 0 relative to the UL pixel origin for 16x mode.		Range: [0,0.9375]	
16x Sample0 Y Offset										
Format:	U0.4									
Subpixel Y offset of Sample 0 relative to the UL pixel origin for 16x mode.										
Range: [0,0.9375]										
2	31:28	<table border="1"> <tr> <td colspan="2">16x Sample7 X Offset</td> </tr> <tr> <td>Format:</td> <td>U0.4</td> </tr> </table>	16x Sample7 X Offset		Format:	U0.4				
16x Sample7 X Offset										
Format:	U0.4									

3DSTATE_SAMPLE_PATTERN

		Subpixel X offset of Sample 7 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
27:24	16x Sample7 Y Offset	Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
23:20	16x Sample6 X Offset	Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_8 or _16. Range: [0,0.9375]
19:16	16x Sample6 Y Offset	Format: U0.4 Subpixel Y offset of Sample 6 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
15:12	16x Sample5 X Offset	Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
11:8	16x Sample5 Y Offset	Format: U0.4 Subpixel Y offset of Sample 5 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_8 or _16. Range: [0,0.9375]
7:4	16x Sample4 X Offset	Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_8 or _16. Range: [0,0.9375]
3:0	16x Sample4 Y Offset	Format: U0.4

3DSTATE_SAMPLE_PATTERN		
		<p>Subpixel Y offset of Sample 4 relative to the UL pixel origin for 16x mode.</p> <p>Range: [0,0.9375]</p>
3	31:28	<p>16x Sample11 X Offset</p> <p>Format: U0.4</p> <p>Subpixel X offset of Sample 11 relative to the UL pixel origin for 16x mode.</p> <p>Range: [0,0.9375]</p>
		<p>16x Sample11 Y Offset</p> <p>Format: U0.4</p> <p>Subpixel Y offset of Sample 11 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_16.</p> <p>Range: [0,0.9375]</p>
	23:20	<p>16x Sample10 X Offset</p> <p>Format: U0.4</p> <p>Subpixel X offset of Sample 10 relative to the UL pixel origin for 16x mode.</p> <p>Range: [0,0.9375]</p>
		<p>16x Sample10 Y Offset</p> <p>Format: U0.4</p> <p>Subpixel Y offset of Sample 10 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_16</p> <p>Range: [0,0.9375]</p>
	15:12	<p>16x Sample9 X Offset</p> <p>Format: U0.4</p> <p>Subpixel X offset of Sample 9 relative to the UL pixel origin for 16x mode.</p> <p>Range: [0,0.9375]</p>
		<p>16x Sample9 Y Offset</p> <p>Format: U0.4</p> <p>Subpixel Y offset of Sample 9 relative to the UL pixel origin for 16x mode.</p> <p>Range: [0,0.9375]</p>
	7:4	<p>16x Sample8 X Offset</p> <p>Format: U0.4</p> <p>Subpixel X offset of Sample 8 relative to the UL pixel origin for 16x mode.</p>

3DSTATE_SAMPLE_PATTERN		
		Range: [0,0.9375]
	3:0	16x Sample8 Y Offset Format: U0.4 Subpixel Y offset of Sample 8 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
4	31:28	16x Sample15 X Offset Format: U0.4 Subpixel X offset of Sample 15 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	16x Sample15 Y Offset Format: U0.4 Subpixel Y offset of Sample 15 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	23:20	16x Sample14 X Offset Format: U0.4 Subpixel X offset of Sample 14 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_16. Range: [0,0.9375]
	19:16	16x Sample14 Y Offset Format: U0.4 Subpixel Y offset of Sample 14 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_16 Range: [0,0.9375]
	15:12	16x Sample13 X Offset Format: U0.4 Subpixel X offset of Sample 13 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	11:8	16x Sample13 Y Offset Format: U0.4 Subpixel Y offset of Sample 13 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	7:4	16x Sample12 X Offset Format: U0.4 Subpixel X offset of Sample 12 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_16. Range: [0,0.9375]
	3:0	16x Sample12 Y Offset Format: U0.4 Subpixel Y offset of Sample 12 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
5	31:28	8x Sample7 X Offset Format: U0.4 Subpixel X offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	8x Sample7 Y Offset Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	8x Sample6 X Offset Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	8x Sample6 Y Offset Format: U0.4 Subpixel Y offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	8x Sample5 X Offset Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	8x Sample5 Y Offset Format: U0.4

3DSTATE_SAMPLE_PATTERN		
		Subpixel Y offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	8x Sample4 X Offset Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	8x Sample4 Y Offset Format: U0.4 Subpixel Y offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
6	31:28	8x Sample3 X Offset Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	8x Sample3 Y Offset Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	8x Sample2 X Offset Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	8x Sample2 Y Offset Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	8x Sample1 X Offset Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	11:8	8x Sample1 Y Offset
		Format: U0.4
		Subpixel Y offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	8x Sample0 X Offset
		Format: U0.4
		Subpixel X offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	8x Sample0 Y Offset
		Format: U0.4
		Subpixel Y offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
7	31:28	4x Sample3 X Offset
		Format: U0.4
		Subpixel X offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	27:24	4x Sample3 Y Offset
		Format: U0.4
		Subpixel Y offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	23:20	4x Sample2 X Offset
		Format: U0.4
		Subpixel X offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	19:16	4x Sample2 Y Offset
		Format: U0.4
		Subpixel Y offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	15:12	4x Sample1 X Offset
		Format: U0.4

3DSTATE_SAMPLE_PATTERN			
		Subpixel X offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]	
	11:8	4x Sample1 Y Offset Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]	
	7:4	4x Sample0 X Offset Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]	
	3:0	4x Sample0 Y Offset Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]	
	8	31:24	Reserved Format: MBZ
		23:20	1x Sample0 X Offset Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]
		19:16	1x Sample0 Y Offset Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]
		15:12	2x Sample1 X Offset Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
		11:8	2x Sample1 Y Offset Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 2x mode.

3DSTATE_SAMPLE_PATTERN	
	Range: [0,0.9375]
7:4	2x Sample0 X Offset
	Format: U0.4
	Subpixel X offset of Sample 0 relative to the UL pixel origin for 2x mode.
	Range: [0,0.9375]
3:0	2x Sample0 Y Offset
	Format: U0.4
	Subpixel Y offset of Sample 0 relative to the UL pixel origin for 2x mode.
	Range: [0,0.9375]

3DSTATE_SAMPLER_STATE_POINTERS_DS

3DSTATE_SAMPLER_STATE_POINTERS_DS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_SAMPLER_STATE_POINTERS_DS command is used to define the location of DS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	2Dh 3DSTATE_SAMPLER_STATE_POINTERS_DS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Sampler State Pointers State Body	
		Format:	3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_GS

3DSTATE_SAMPLER_STATE_POINTERS_GS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_SAMPLER_STATE_POINTERS_GS command is used to define the location of GS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	2Eh 3DSTATE_SAMPLER_STATE_POINTERS_GS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Sampler State Pointers State Body	
		Format:	3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_HS

3DSTATE_SAMPLER_STATE_POINTERS_HS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_SAMPLER_STATE_POINTERS_HS command is used to define the location of HS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	2Ch 3DSTATE_SAMPLER_STATE_POINTERS_HS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Sampler State Pointers State Body	
		Format:	3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_PS

3DSTATE_SAMPLER_STATE_POINTERS_PS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_SAMPLER_STATE_POINTERS_PS command is used to define the location of PS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	2Fh 3DSTATE_SAMPLER_STATE_POINTERS_PS	
	Format:	OpCode	
15	POSH-Enabled Render State Optimization Enable 1		
14:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Sampler State Pointers State Body	
		Format:	3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_VS

3DSTATE_SAMPLER_STATE_POINTERS_VS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_SAMPLER_STATE_POINTERS_VS command is used to define the location of VS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	2Bh 3DSTATE_SAMPLER_STATE_POINTERS_VS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Sampler State Pointers State Body	
		Format:	3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SBE

3DSTATE_SBE		
Source:	RenderCS	
Length Bias:	2	
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
		Format: OpCode
	26:24	3D Command Opcode
Default Value: 0h 3DSTATE_PIPELINED		
Format: OpCode		
23:16	3D Command Sub Opcode	
	Default Value: 1Fh 3DSTATE_SBE	
	Format: OpCode	
15:8	Reserved	
	Format: MBZ	
7:0	7:0	DWord Length
		Default Value: 04h Excludes DWord (0,1)
		Format: =n
		Total Length - 2
1..5	159:0	SBE State Body
		Format: 3DSTATE_SBE_BODY

3DSTATE_SBE_SWIZ

3DSTATE_SBE_SWIZ			
Source:	RenderCS		
Length Bias:	2		
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		51h 3DSTATE_SBE_SWIZ	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	9h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..10	319:0	SBE SWIZ State Body	
		Format:	3DSTATE_SBE_SWIZ_BODY



3DSTATE_SCISSOR_STATE_POINTERS

3DSTATE_SCISSOR_STATE_POINTERS			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_SCISSOR_STATE_POINTERS command is used to define the location of the indirect SCISSOR_RECT state.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		0Fh 3DSTATE_SCISSOR_STATE_POINTERS	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Scissor State Pointers Body	
		Format:	3DSTATE_SCISSOR_STATE_POINTERS_BODY

3DSTATE_SF

3DSTATE_SF			
Source:	RenderCS		
Length Bias:	2		
Restriction			
When executed in the POCS command stream, this command programs the state for the SFR stage of the POCS pipeline.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		13h 3DSTATE_SF	
Format:		OpCode	
15:11	Reserved		
	Format:	MBZ	
10	SF State Modify Disable		
	Format:	Disable	
When this bit is set, the following fields will be ignored: <ul style="list-style-type: none"> • Legacy Global Depth Bias Enable • Statistics Enable • Viewport Transform Enable • Fast Scissor Clip Disable • Line End Cap Antialiasing Region Width • Zero Pixel Triangle Filter Disable • 2x2 Pixel Triangle Filter Disable • Last Pixel Enable • Triangle Strip/List Provoking Vertex Select • Line Strip/List Provoking Vertex Select • Triangle Fan Provoking Vertex Select 			

3DSTATE_SF						
		<ul style="list-style-type: none"> • AA Line Distance Mode • Smooth Point Enable • Vertex Sub Pixel Precision Select • Point Width Source 				
	9	<p>Line Width Modify Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Line Width 	Format:	Disable		
Format:	Disable					
	8	<p>Point Width Modify Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Point Width 	Format:	Disable		
Format:	Disable					
	7:0	<p>DWord Length</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>2h Excludes DWord (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>Total Length - 2</p>	Default Value:	2h Excludes DWord (0,1)	Format:	=n
Default Value:	2h Excludes DWord (0,1)					
Format:	=n					
1..3	95:0	<p>SF State Body</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Format:</td> <td>3DSTATE_SF_BODY</td> </tr> </table>	Format:	3DSTATE_SF_BODY		
Format:	3DSTATE_SF_BODY					

3DSTATE_SLICE_TABLE_STATE_POINTERS

3DSTATE_SLICE_TABLE_STATE_POINTERS							
Source:		RenderCS					
Length Bias:		2					
DWord	Bit	Description					
0	31:29	Command Type					
		Default Value:	3h GFXPIPE				
		Format:	Opcode				
	28:27	Command SubType					
		Default Value:	3h GFXPIPE_3D				
		Format:	Opcode				
	26:24	3D Command Opcode					
Default Value:		1h 3DSTATE_NONPIPELINED					
Format:		OpCode					
23:16	3D Command Sub Opcode						
	Default Value:	20h 3DSTATE_SLICE_TABLE_STATE_POINTERS					
	Format:	OpCode					
15:8	Reserved						
	Format:	MBZ					
7:0	DWord Length						
	Default Value:	0h Excludes DWord (0,1)					
	Format:	=n					
1	31:6	Slice Hash Table State Pointer					
		Format: DynamicStateOffset[31:6]SLICE_HASH_TABLE					
	Specifies the 64-byte aligned offset of the SLICE_HASH_TABLE. This offset is relative to the Dynamic State Base Address .						
5:1	Reserved						
	Format:	MBZ					
0	0	Slice Hash State Pointer Valid					
		Format: Enable					
	This bit, if set, indicates that the SLICE_HASH_TABLE pointer has changed and new state needs to be fetched.						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> </tr> <tr> <td>1h</td> <td>Enable</td> </tr> </tbody> </table>	Value	Name	0h	Disable	1h
Value	Name						
0h	Disable						
1h	Enable						



3DSTATE_SO_BUFFER_INDEX_0

3DSTATE_SO_BUFFER_INDEX_0			
Source:	RenderCS		
Length Bias:	2		
This command is to program the SO buffer addresses and Attributes for Index 0.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		60h 3DSTATE_SO_BUFFER_INDEX_0	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	6h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..7	223:0	SO Buffer Index State Body	
		Format:	3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER_INDEX_1

3DSTATE_SO_BUFFER_INDEX_1			
Source:	RenderCS		
Length Bias:	2		
This command is to program the SO buffer addresses and Attributes for Index 1.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		61h 3DSTATE_SO_BUFFER_INDEX_1	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	6h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..7	223:0	SO Buffer Index State Body	
		Format:	3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER_INDEX_2

3DSTATE_SO_BUFFER_INDEX_2			
Source:	RenderCS		
Length Bias:	2		
This command is to program the SO buffer addresses and Attributes for Index 2.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		62h 3DSTATE_SO_BUFFER_INDEX_2	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	6h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..7	223:0	SO Buffer Index State Body	
		Format:	3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER_INDEX_3

3DSTATE_SO_BUFFER_INDEX_3			
Source:	RenderCS		
Length Bias:	2		
This command is to program the SO buffer addresses and Attributes for Index 3.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		63h 3DSTATE_SO_BUFFER_INDEX_3	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	6h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..7	223:0	SO Buffer Index State Body	
		Format:	3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER

3DSTATE_SO_BUFFER		
Source:	RenderCS	
Length Bias:	2	
Programming Notes		
Foreach SO Buffer, the 3DSTATE_SO_BUFFER must only be sent once prior to each 3DPRIMITIVE command.		
Workaround		
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
		Default Value: 18h 3DSTATE_SO_BUFFER Format: OpCode
	15:8	Reserved
		Format: MBZ
7:0	DWord Length	
	Default Value: 6h Excludes DWord (0,1) Format: =n	
	Total Length - 2	
1	31	SO Buffer Enable
		Format: Enable If set, stream output to SO Buffer is enabled, , if 3DSTATE_STREAMOUT::SO Function ENABLE is also enabled. If clear, the SO Buffer is considered "not bound" and effectively treated as a zero-length buffer for the purposes of SO output and overflow detection. If an enabled stream's Stream to Buffer Selects includes this buffer it is by definition an overflow condition. That stream will cause no writes to occur, and only SO_PRIM_STORAGE_NEEDED[<stream>] will increment.
	30:29	SO Buffer Index
		Format: U2

3DSTATE_SO_BUFFER	
	Specifies which of the four SO Buffers is being defined.
28:22	<p>SO Buffer Object Control State</p> <p>Format: MEMORY_OBJECT_CONTROL_STATE</p> <p>Specifies the memory object control state for the SO buffer.</p>
21	<p>Stream Offset Write Enable</p> <p>Format: Enable</p> <p>When set, this field allows the hardware to write SO_WRITE_OFFSET[Buffer#] as specified in the Stream Offset field.</p> <p style="text-align: center;">Programming Notes</p> <p>The field is operates irrespective of whether SO Buffer Enable is set or clear.</p>
20	<p>Stream Output Buffer Offset Address Enable</p> <p>Format: Enable</p> <p>When set, this field allows the hardware to read/write the stream output buffer offset as specified in the "Stream Output Buffer Offset Address" field.</p> <p style="text-align: center;">Programming Notes</p> <p>The field is operates irrespective of whether SO Buffer Enable is set or clear.</p>
19:0	<p>Reserved</p> <p>Format: MBZ</p>
2..3	<p>63:2 Surface Base Address</p> <p>Format: VIRTUAL_ADDR[63:2]</p> <p>This field specifies the starting address of the buffer in Graphics Memory.</p>
	<p>1:0 Reserved</p> <p>Format: MBZ</p>
4	<p>31:30 Reserved</p> <p>Format: MBZ</p>
	<p>29:0 Surface Size</p> <p>Format: U30-1</p> <p>This field specifies the size of buffer in number DWords minus 1 of the buffer in Graphics Memory.</p>
5..6	<p>63:2 Stream Output Buffer Offset Address</p> <p>Format: VIRTUAL_ADDR[63:2]</p> <p>This field specifies the starting address of the buffer in Graphics Memory where the Stream Output Buffer Offset is stored when all the data has been written. It is also used to fetch the stream Output buffer Offset when needed.</p>

3DSTATE_SO_BUFFER				
	1:0	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
7	31:0	Stream Offset This field specifies the Offset in stream output buffer to start at, or whether to append to the end of an existing buffer. The Offset must be DWORD aligned. If Stream Offset is equal to 0xFFFFFFFF then load the value at the Stream Output Buffer Offset address into SO_WRITE_OFFSET[Buffer#]. Otherwise, SO_WRITE_OFFSET[Buffer#] = Stream Offset.		

3DSTATE_SO_DECL_LIST

3DSTATE_SO_DECL_LIST			
Source:	RenderCS		
Length Bias:	2		
Workaround			
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	1h 3DSTATE_NONPIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		17h 3DSTATE_SO_DECL_LIST	
Format:		OpCode	
15:9	Reserved		
	Format:	MBZ	
8:0	DWord Length	Format:	=n
		Value	Name
	[1,257]	Excludes DWORD (0,1) 0-128 Entries	Value = 2 * (# of SO_DECL quads) + 1
1	31:16	Reserved	
		Format:	MBZ
	15:12	Stream to Buffer Selects [3]	
		Format:	U4
		Identifies to which SO Buffers stream 3 outputs. See Stream To Buffer Selects [0] field description.	
Value		Name	
1xxb	SO Buffer 3		
x1xxb	SO Buffer 2		
xx1xb	SO Buffer 1		

3DSTATE_SO_DECL_LIST											
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">xxx1b</td> <td>SO Buffer 0</td> </tr> </table>	xxx1b	SO Buffer 0								
xxx1b	SO Buffer 0										
11:8	Stream to Buffer Selects [2] Format: U4 Identifies to which SO Buffers stream 2 outputs. See Stream To Buffer Selects [0] field description.										
	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td>1xxx</td> <td>SO Buffer 3</td> </tr> <tr> <td>x1xx</td> <td>SO Buffer 2</td> </tr> <tr> <td>xx1x</td> <td>SO Buffer 1</td> </tr> <tr> <td>xxx1</td> <td>SO Buffer 0</td> </tr> </tbody> </table>	Value	Name	1xxx	SO Buffer 3	x1xx	SO Buffer 2	xx1x	SO Buffer 1	xxx1	SO Buffer 0
	Value	Name									
	1xxx	SO Buffer 3									
	x1xx	SO Buffer 2									
xx1x	SO Buffer 1										
xxx1	SO Buffer 0										
7:4	Stream to Buffer Selects [1] Format: U4 Identifies to which SO Buffers stream 1 outputs. See Stream To Buffer Selects [0] field description.										
3:0	Stream to Buffer Selects [0] Format: U4 Identifies to which SO Buffers stream 0 outputs (irrespective of whether those buffers are enabled via 3DSTATE_STREAMOUT). Software is required to scan the SO_DECL list in order to provide this summary information. Note: For "inactive" streams, software must program this field to all zero (no buffers written to) and the corresponding Num Entries field to zero (no valid SO_DECLS).										
2	31:24 Num Entries [3] Format: U8 Specifies the number of valid SO_DECL entries for Stream 3. (See notes in Num Entries [0] field description).										
	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,128]</td> <td>entries</td> </tr> </tbody> </table>	Value	Name	[0,128]	entries						
Value	Name										
[0,128]	entries										
23:16	Num Entries [2]										

3DSTATE_SO_DECL_LIST											
	<table border="1"> <tr> <td>Format:</td> <td>U8</td> </tr> <tr> <td colspan="2">Specifies the number of valid SO_DECL entries for Stream 2. (See notes in Num Entries [0] field description).</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td>[0,128]</td> <td>entries</td> </tr> </table>	Format:	U8	Specifies the number of valid SO_DECL entries for Stream 2. (See notes in Num Entries [0] field description).		Value	Name	[0,128]	entries		
Format:	U8										
Specifies the number of valid SO_DECL entries for Stream 2. (See notes in Num Entries [0] field description).											
Value	Name										
[0,128]	entries										
15:8	<table border="1"> <tr> <td colspan="2">Num Entries [1]</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> <tr> <td colspan="2">Specifies the number of valid SO_DECL entries for Stream 1. (See notes in Num Entries [0] field description).</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td>[0,128]</td> <td>entries</td> </tr> </table>	Num Entries [1]		Format:	U8	Specifies the number of valid SO_DECL entries for Stream 1. (See notes in Num Entries [0] field description).		Value	Name	[0,128]	entries
Num Entries [1]											
Format:	U8										
Specifies the number of valid SO_DECL entries for Stream 1. (See notes in Num Entries [0] field description).											
Value	Name										
[0,128]	entries										
7:0	<table border="1"> <tr> <td colspan="2">Num Entries [0]</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> <tr> <td colspan="2">Specifies the number of valid SO_DECL entries for Stream 0. Note that the SO_DECLs are programmed in groups of four (one SO_DECL for each of the four streams). Therefore the number of 2-DWord groups of SO_DECLs supplied in this command is derived from the stream(s) with the most valid SO_DECLs. The NumEntries value specific to each stream will indicate how many SO_DECLs are valid for that particular stream. Any trailing invalid SO_DECLs supplied for streams with fewer valid SO_DECLs will be ignored. It is legal to specify Num Entries = 0 for all four streams simultaneously. In this case there will be no SO_DECLs included in the command (only DW 0-2). Note that all Stream to Buffer Selects bits must be zero in this case (as no streams produce output).</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td>[0,128]</td> <td>entries</td> </tr> </table>	Num Entries [0]		Format:	U8	Specifies the number of valid SO_DECL entries for Stream 0. Note that the SO_DECLs are programmed in groups of four (one SO_DECL for each of the four streams). Therefore the number of 2-DWord groups of SO_DECLs supplied in this command is derived from the stream(s) with the most valid SO_DECLs. The NumEntries value specific to each stream will indicate how many SO_DECLs are valid for that particular stream. Any trailing invalid SO_DECLs supplied for streams with fewer valid SO_DECLs will be ignored. It is legal to specify Num Entries = 0 for all four streams simultaneously. In this case there will be no SO_DECLs included in the command (only DW 0-2). Note that all Stream to Buffer Selects bits must be zero in this case (as no streams produce output).		Value	Name	[0,128]	entries
Num Entries [0]											
Format:	U8										
Specifies the number of valid SO_DECL entries for Stream 0. Note that the SO_DECLs are programmed in groups of four (one SO_DECL for each of the four streams). Therefore the number of 2-DWord groups of SO_DECLs supplied in this command is derived from the stream(s) with the most valid SO_DECLs. The NumEntries value specific to each stream will indicate how many SO_DECLs are valid for that particular stream. Any trailing invalid SO_DECLs supplied for streams with fewer valid SO_DECLs will be ignored. It is legal to specify Num Entries = 0 for all four streams simultaneously. In this case there will be no SO_DECLs included in the command (only DW 0-2). Note that all Stream to Buffer Selects bits must be zero in this case (as no streams produce output).											
Value	Name										
[0,128]	entries										
3..n	<table border="1"> <tr> <td>63:0</td> <td>Entry</td> </tr> <tr> <td>Format:</td> <td>SO_DECL_ENTRY</td> </tr> </table>	63:0	Entry	Format:	SO_DECL_ENTRY						
63:0	Entry										
Format:	SO_DECL_ENTRY										

3DSTATE_STENCIL_BUFFER

3DSTATE_STENCIL_BUFFER			
Source:	RenderCS		
Length Bias:	2		
<p>The stencil buffer surface state is delivered as a pipelined state packet. However, the state change pipelining isn't completely transparent (see restriction below).</p> <p>WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.</p>			
Programming Notes			
If the Stencil surface is not present, SW must set the Surface Type field to SURFTYPE_NULL.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	6h 3DSTATE_STENCIL_BUFFER	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length	Default Value:	6h Excludes Dword (0,1)
		Format:	=n
		Excludes DWord(0,1)	
1..7	223:0	Stencil Buffer State Body	
		Format:	3DSTATE_STENCIL_BUFFER_BODY

3DSTATE_STREAMOUT

3DSTATE_STREAMOUT		
Source:	RenderCS	
Length Bias:	2	
This command contains pipelined state required by the SOL unit.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
Default Value: 1Eh 3DSTATE_STREAMOUT Format: OpCode		
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 3h Excludes DWord (0,1) Format: =n	
1..4	127:0	Streamout State Body
		Format: 3DSTATE_STREAMOUT_BODY

3DSTATE_SUBSLICE_HASH_TABLE

3DSTATE_SUBSLICE_HASH_TABLE											
Source:	RenderCS										
Length Bias:	2										
Programmable DualSubSlice hashing control and tables. First DW indicates the mode how the last 4DW are configured and selects the programmable dualsubslice hashing mode for each slice.											
Programming Notes											
All slice references are physical slice. Instruction must be programmed based on which slices and subslices are enabled.											
DWord	Bit	Description									
0	31:29	Command Type									
		Default Value:	3h GFXPIPE								
		Format:	Opcode								
	28:27	Command SubType									
		Default Value:	3h GFXPIPE_3D								
		Format:	Opcode								
	26:24	3D Command Opcode									
Default Value:		1h 3DSTATE_NONPIPELINED									
Format:		OpCode									
23:16	3D Command Sub Opcode										
	Default Value:	1Fh 3DSTATE_SUBSLICE_HASH_TABLE									
	Format:	OpCode									
15:8	Reserved										
	Format:	MBZ									
7:0	DWord Length										
	Format:	=n									
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>Ch</td> <td>Excludes DWord (0,1) [Default]</td> </tr> </tbody> </table>	Value	Name	Ch	Excludes DWord (0,1) [Default]						
Value	Name										
Ch	Excludes DWord (0,1) [Default]										
1	31:30	TableMode									
		Format:	U2								
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Single table [Default]</td> <td>DW2-5 is Table[0] - a single 2-way 128 entry [Y][X] table.</td> </tr> <tr> <td>1h</td> <td>Dual tables</td> <td>DW2-3 is 'Table[0]' and is 2-way 64 entry [Y][X] table. DW4-5 is 'Table[1]' and is 2-way 64 entry [Y][X] table.</td> </tr> </tbody> </table>		Value	Name	Description	0h	Single table [Default]	DW2-5 is Table[0] - a single 2-way 128 entry [Y][X] table.	1h	Dual tables	DW2-3 is 'Table[0]' and is 2-way 64 entry [Y][X] table. DW4-5 is 'Table[1]' and is 2-way 64 entry [Y][X] table.
	Value	Name	Description								
	0h	Single table [Default]	DW2-5 is Table[0] - a single 2-way 128 entry [Y][X] table.								
1h	Dual tables	DW2-3 is 'Table[0]' and is 2-way 64 entry [Y][X] table. DW4-5 is 'Table[1]' and is 2-way 64 entry [Y][X] table.									

3DSTATE_SUBSLICE_HASH_TABLE			
	29:16	Reserved Format: MBZ	
	15:0	SliceHashCtrl PerSlice[7:0]SliceHashControl	
2..5	127:64	64 Entry 2-way Table[1] Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: pixel_hash_table_1bit_64entry	
		128 Entry 2-way Table[0] Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Single Format: pixel_hash_table_1bit_128entry	
	63:0	64 Entry 2-way Table[0] Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: pixel_hash_table_1bit_64entry	
		64 Entry 3-way Table[1] Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: pixel_hash_table_2bit_64entry	
	6..13	255:128	128 Entry 3-way Table[0] Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Single Format: pixel_hash_table_2bit_128entry
			64 Entry 3-way Table[0] Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: pixel_hash_table_2bit_64entry

3DSTATE_TE

3DSTATE_TE				
Source:	RenderCS			
Length Bias:	2			
The state used by TE is defined with this inline state packet.				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value: 3h GFXPIPE Format: OpCode		
	28:27	Command SubType		
		Default Value: 3h GFXPIPE_3D Format: OpCode		
	26:24	3D Command Opcode		
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode		
	23:16	3D Command Sub Opcode		
Default Value: 1Ch 3DSTATE_TE Format: OpCode				
15:8	Reserved			
	Format: MBZ			
7:0	DWord Length			
	Format: =n			
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3h</td> <td>Excludes DWord (0,1) [Default]</td> </tr> </tbody> </table>	Value	Name	3h
Value	Name			
3h	Excludes DWord (0,1) [Default]			
1..4	127:0	TE State Body		
		Format: 3DSTATE_TE_BODY		

3DSTATE_URB_ALLOC_DS

3DSTATE_URB_ALLOC_DS			
Source:	RenderCS		
Length Bias:	2		
<p>When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the DS pipeline stage.</p> <p>Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Hardware will use those values to automatically compute the URB allocation within the total URB space based on the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.</p>			
Programming Notes			
SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.			
SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.			
If Domain Shader Thread Dispatch is Enabled then the minimum number of handles that must be allocated is 50 URB entries.			
SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	5Ah 3DSTATE_URB_ALLOC_DS
		Format:	OpCode
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		

3DSTATE_URB_ALLOC_DS			
		Default Value:	1h DWORD_COUNT_n
		Format:	=n
1..2	63:0	URB Alloc DS State Body	
		Format:	3DSTATE_URB_ALLOC_DS_BODY

3DSTATE_URB_ALLOC_GS

3DSTATE_URB_ALLOC_GS			
Source:	RenderCS		
Length Bias:	2		
<p>When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the GS pipeline stage.</p> <p>Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.</p>			
Programming Notes			
SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.			
SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.			
SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	5Bh 3DSTATE_URB_ALLOC_GS	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	1h DWORD_COUNT_n	
	Format:	=n	
1..2	63:0	URB Alloc GS State Body	

3DSTATE_URB_ALLOC_GS	
Format:	3DSTATE_URB_ALLOC_GS_BODY

3DSTATE_URB_ALLOC_HS

3DSTATE_URB_ALLOC_HS		
Source:	RenderCS	
Length Bias:	2	
<p>When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the HS pipeline stage.</p> <p>Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Hardware will use those values to automatically compute the URB allocation within the total URB space based on the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.</p>		
Programming Notes		
SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.		
SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.		
SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
		Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED
		Format: OpCode
	23:16	3D Command Sub Opcode
		Default Value: 59h 3DSTATE_URB_ALLOC_HS
		Format: OpCode
	15:8	Reserved
		Format: MBZ
7:0	DWord Length	
	Default Value: 1h DWORD_COUNT_n	

3DSTATE_URB_ALLOC_HS		
		Format: =n
1..2	63:0	URB Alloc HS State Body Format: 3DSTATE_URB_ALLOC_HS_BODY

3DSTATE_URB_ALLOC_VS

3DSTATE_URB_ALLOC_VS			
Source:	RenderCS, PositionCS		
Length Bias:	2		
<p>When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the VF and VS pipeline stages.</p>			
<p>When executed from the POCS command stream, this command provides the state variables associated with the URB region used by the VFR and VSR pipeline stages. The POCS command stream will only execute 3DSTATE_URB_ALLOC_VS command with respect to URB programming for the POCS 3D pipeline. 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS and 3DSTATE_URB_ALLOC_GS commands are ignored by the POCS command stream.</p>			
<p>Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Hardware will use those values to automatically compute the URB allocation within the total URB space based on the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.</p>			
Programming Notes			
<p>SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.</p>			
<p>The VSR URB region shall never overlap any other URB region. As POCS and RCS command streams are not implicitly synchronized, if POCS is used SW shall reserve a region of the URB for use by VSR. Both pipelines must be flushed and synchronized before expanding the VSR URB region such that the new VSR URB region overlaps URB space previously used by the render pipeline, or the URB space to be used by the render pipeline overlaps the previous VSR URB region.</p>			
<p>When specifying URB allocation state for the RCS 3D pipe, SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.</p>			
<p>SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
26:24	3D Command Opcode		

3DSTATE_URB_ALLOC_VS			
		Default Value: 0h 3DSTATE_PIPELINED	
		Format: OpCode	
	23:16	3D Command Sub Opcode	
		Default Value: 58h 3DSTATE_URB_ALLOC_VS	
		Format: OpCode	
		Reserved	
	15:8	Format: MBZ	
	7:0	DWord Length	
		Default Value: 1h DWORD_COUNT_n	
		Format: =n	
1..2	63:0	URB Alloc VS State Body	
		Format: 3DSTATE_URB_ALLOC_VS_BODY	

3DSTATE_URB_DS

3DSTATE_URB_DS		
Source:	RenderCS	
Length Bias:	2	
<p>This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.</p> <p>The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations.</p>		
Programming Notes		
<p>When programming DS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_HS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.</p> <p>Please see 3DSTATE_URB_ALLOC_DS for any new programming notes related to URB programming.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED
23:16	3D Command Sub Opcode	
	Default Value: 32h 3DSTATE_URB_DS	
15:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 0h DWORD_COUNT_n	
1	31:0	URB DS State Body
		Format: 3DSTATE_URB_DS_BODY

3DSTATE_URB_GS

3DSTATE_URB_GS			
Source:	RenderCS		
Length Bias:	2		
<p>This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.</p> <p>The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations</p>			
Programming Notes			
<p>When programming GS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_HS, and 3DSTATE_URB_DS must also be programmed in order for the programming of this state to be valid.</p> <p>Please see 3DSTATE_URB_ALLOC_GS for any new programming notes related to URB programming.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
23:16	3D Command Sub Opcode		
	Default Value:	33h 3DSTATE_URB_GS	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
1	31:0	URB GS State Body	
		Format:	3DSTATE_URB_GS_BODY

3DSTATE_URB_HS

3DSTATE_URB_HS		
Source:	RenderCS	
Length Bias:	2	
<p>This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.</p> <p>The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations</p>		
Programming Notes		
<p>When programming HS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_DS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.</p> <p>Please see 3DSTATE_URB_ALLOC_HS for any new programming notes related to URB programming.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
23:16	3D Command Sub Opcode	
	Default Value: 31h 3DSTATE_URB_HS Format: OpCode	
15:8	Reserved	
7:0	Format: MBZ	DWord Length
		Default Value: 0h DWORD_COUNT_n
		Format: =n
1	31:0	URB HS State Body
		Format: 3DSTATE_URB_HS_BODY

3DSTATE_URB_VS

3DSTATE_URB_VS			
Source:	RenderCS, PositionCS		
Length Bias:	2		
Description			
VS URB Entry Allocation Size equal to 4(5 512-bit URB rows) may cause performance to decrease due to banking in the URB. Element sizes of 16 to 20 should be programmed with six 512-bit URB rows.			
This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.			
The offset and size should be programmed as if there is only one slice enabled. Hardware will grow the size based on the slice configuration. Software shall ensure that the values programmed do not exceed the URB capacity of one slice. Refer to the L3 allocation and programming guide for valid URB configurations.			
When executed from the POCS command stream, state for the VSR stage is set. The VSR URB region shall never overlap any other URB region. As POCS and RCS command streams are not implicitly synchronized, if POCS is used SW shall reserve a region of the URB for use by VSR. Both pipelines must be flushed and synchronized before expanding the VSR URB region such that the new VSR URB region overlaps URB space previously used by the render pipeline, or the URB space to be used by the render pipeline overlaps the previous VSR URB region. The POCS command stream will only execute 3DSTATE_URB_VS command with respect to URB programming for the POCS 3D pipeline. 3DSTATE_URB_HS, 3DSTATE_URB_DS and 3DSTATE_URB_GS commands are ignored by the POCS command stream.			
Programming Notes			
When programming VS URB state for the RCS 3D pipe, 3DSTATE_URB_HS, 3DSTATE_URB_DS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.			
Please see 3DSTATE_URB_ALLOC_VS for any new programming notes related to URB programming.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	30h 3DSTATE_URB_VS
		Format:	OpCode

3DSTATE_URB_VS		
	15:8	Reserved Format: MBZ
	7:0	DWord Length Default Value: 0h DWORD_COUNT_n Format: =n
	1	URB VS State Body Format: 3DSTATE_URB_VS_BODY

3DSTATE_VERTEX_BUFFERS

3DSTATE_VERTEX_BUFFERS			
Source:	RenderCS		
Length Bias:	2		
This command is used to specify VB state used by the VF function.			
Can specify from 1 to 33 VBs.			
The VertexBufferID field within a VERTEX_BUFFER_STATE structure indicates the specific VB. If a VB definition is not included in this command, its associated state is left unchanged and is available for use if previously defined.			
Programming Notes			
It is possible to have individual vertex elements sourced completely from generated ID values and therefore not require any vertex buffer accesses for that vertex element. In this case, VFfunction will simply ignore the VB state associated with that vertex element. If all enabled vertex elements have this characteristic, no VBs are required to process 3DPRIMITIVE commands. For example, this might arise when the user wants to perform all data lookups in the first shader, so only generated index values need to be passed down to it. In this extreme case, SW would not need to program any VB state, and therefore not need to issue any 3DSTATE_VERTEX_BUFFERS commands.			
For any 3DSTATE_VERTEX_BUFFERS command, at least one VERTEX_BUFFER_STATE structure must be included.			
VERTEX_BUFFER_STATE structures are 4 DWords for both VERTEXDATA buffers and INSTANCEDATA buffers.			
Inclusion of partial VERTEX_BUFFER_STATE structures is UNDEFINED.			
The order in which VBs are defined within this command can be arbitrary, though a vertex buffer must be defined only once in any given command (otherwise operation is UNDEFINED).			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	03h GFXPIPE
		Format:	Opcode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	Opcode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	Opcode
	23:16	3D Command Sub Opcode	
		Default Value:	08h 3DSTATE_VERTEX_BUFFERS
		Format:	Opcode

3DSTATE_VERTEX_BUFFERS								
	15	Reserved Format: _____ MBZ						
	14:8	Reserved Format: _____ MBZ						
	7:0	DWord Length Format: _____ =n n = 4b-1 (where b = # of buffer states included)						
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td>DWORD_COUNT_n [Default]</td> </tr> <tr> <td style="text-align: center;">[3,131]</td> <td>1-33 Buffers</td> </tr> </tbody> </table>	Value	Name	3	DWORD_COUNT_n [Default]	[3,131]	1-33 Buffers
	Value	Name						
	3	DWORD_COUNT_n [Default]						
[3,131]	1-33 Buffers							
1..n	127:0	Vertex Buffer State Format: _____ VERTEX_BUFFER_STATE						

3DSTATE_VERTEX_ELEMENTS

3DSTATE_VERTEX_ELEMENTS			
Source:	RenderCS		
Length Bias:	2		
<p>This is a variable-length command used to specify the active vertex elements. Each VERTEX_ELEMENT_STATE structure contains a Valid bit which determines which elements are used. Any elements not programmed by this command are disabled.</p>			
Up to 34 elements.			
Programming Notes			
At least one VERTEX_ELEMENT_STATE structure must be included.			
Inclusion of partial VERTEX_ELEMENT_STATE structures is UNDEFINED.			
SW must ensure that at least one vertex element is defined prior to issuing a 3DPRIMITIVE command, or operation is UNDEFINED.			
There are no 'holes' allowed in the destination vertex: NOSTORE components must be overwritten by subsequent components unless they are the trailing DWords of the vertex. Software must explicitly chose some value (probably 0) to be written into DWords that would otherwise be 'holes'.			
Within a VERTEX_ELEMENT_STATE structure, if a Component Control field is set to something other than VFCOMP_STORE_SRC, no higher-numbered Component Control fields may be set to VFCOMP_STORE_SRC. In other words, only trailing components can be set to something other than VFCOMP_STORE_SRC.			
See additional restrictions listed in the command fields and VERTEX_ELEMENT_STATE description.			
Element[0] must be valid.			
All elements must be valid from Element[0] to the last valid element. (E.g.. if Element[2] is valid then Element[1] and Element[0] must also be valid).			
The pitch between elements packed in the URB will always be 128 bits.			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	03h GFXPIPE
		Format:	Opcode
	28:27	Command SubType	
		Default Value:	3h 3D
		Format:	Opcode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	Opcode
	23:16	3D Command Sub Opcode	

3DSTATE_VERTEX_ELEMENTS													
		<table border="1"> <tr> <td>Default Value:</td> <td>09h 3DSTATE_VERTEX_ELEMENTS</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	09h 3DSTATE_VERTEX_ELEMENTS	Format:	Opcode							
Default Value:	09h 3DSTATE_VERTEX_ELEMENTS												
Format:	Opcode												
	15	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	14:8	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	7:0	DWord Length <table border="1"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>Vertex Element Count = (DWord Count + 1) / 2</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>DWORD_COUNT_n [Default]</td> <td>excludes DWords 0,1</td> </tr> <tr> <td>[1,67]</td> <td>Range</td> <td>1-34 Elements</td> </tr> </tbody> </table>	Format:	=n	Value	Name	Description	1	DWORD_COUNT_n [Default]	excludes DWords 0,1	[1,67]	Range	1-34 Elements
Format:	=n												
Value	Name	Description											
1	DWORD_COUNT_n [Default]	excludes DWords 0,1											
[1,67]	Range	1-34 Elements											
1..n	63:0	Element <table border="1"> <tr> <td>Format:</td> <td>VERTEX_ELEMENT_STATE</td> </tr> </table>	Format:	VERTEX_ELEMENT_STATE									
Format:	VERTEX_ELEMENT_STATE												

3DSTATE_VF_COMPONENT_PACKING

3DSTATE_VF_COMPONENT_PACKING			
Source:	RenderCS		
Length Bias:	2		
<p>This command is used to specify, separately for Vertex Elements [0-31], which post-conversion, 32-bit components are "enabled" to be stored in the URB, and which are "disabled" (not stored). 128 per-component enable bits are provided. Disabling all four components for a given Vertex Element will result in no data stored for that element. Note that any insertion of SGVs (3DSTATE_VF_SGVS) is performed before the packing operation. The Component Packing Enable bit (3DSTATE_VF) controls the overall packing process. If that bit is set, the packing process is enabled and the bit mask provided in this command is used to control which components are stored. If that bit is clear, the packing process is disabled - all four components of "valid" Vertex Elements will be stored.</p>			
<p>Programming Restrictions:</p> <ul style="list-style-type: none"> The Vertex Elements referenced in this command correspond to the first 32 VERTEX_ELEMENT structures passed in 3DSTATE_VERTEX_ELEMENTS. A Vertex Element must be marked as "Valid" via 3DSTATE_VERTEX_ELEMENTS or be an SGV or an element between the last valid element and the last SGV in order for the corresponding Component Enable bits of this command to be utilized. No enable bits are provided for Vertex Elements [32-33], and therefore no packing is performed on these elements (if Valid, all 4 components are stored). If a Vertex Element has Edge Flag Enable set no packing is performed for that element and the corresponding packing state is ignored. Component packing is probably only useful for SIMD8 VS thread execution. 			
At least one component of one "valid" Vertex Element must be enabled.			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.			
Software shall enable all components (XYZW) for any and all VERTEX_ELEMENTS associated with a 256-bit SURFACE_FORMAT. It is INVALID to disable any components in these cases.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode

3DSTATE_VF_COMPONENT_PACKING						
	23:16	3D Command Sub Opcode <table border="1"> <tr> <td>Default Value:</td> <td>55h 3DSTATE_VF_COMPONENT_PACKING</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	55h 3DSTATE_VF_COMPONENT_PACKING	Format:	OpCode
	Default Value:	55h 3DSTATE_VF_COMPONENT_PACKING				
	Format:	OpCode				
	15	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
14:8	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
	7:0	DWord Length <table border="1"> <tr> <td>Default Value:</td> <td>3h Excludes DWord (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	3h Excludes DWord (0,1)	Format:	=n
Default Value:	3h Excludes DWord (0,1)					
Format:	=n					
1.4	127:0	VF Component Packing State Body <table border="1"> <tr> <td>Format:</td> <td>3DSTATE_VF_COMPONENT_PACKING_BODY</td> </tr> </table>	Format:	3DSTATE_VF_COMPONENT_PACKING_BODY		
Format:	3DSTATE_VF_COMPONENT_PACKING_BODY					

3DSTATE_VF_INSTANCING

3DSTATE_VF_INSTANCING			
Source:	RenderCS		
Length Bias:	2		
This command is used to control the "instancing" state associated with a specific vertex element.			
Programming Notes			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		49h 3DSTATE_VF_INSTANCING	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Format:	=n	
	Value	Name	
	1h	Excludes DWord (0,1) [Default]	
	43h	Context Restore	
1..2	63:0	VF Instancing State Body	
		Format:	3DSTATE_VF_INSTANCING_BODY

3DSTATE_VF

3DSTATE_VF			
Source:	RenderCS		
Length Bias:	2		
This command is used to set various state variables in the VF stage.			
The use of the component packing mask is specified via 3DSTATE_VF_COMPONENT_PACKING			
Programming Notes			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	0Ch 3DSTATE_VF
		Format:	OpCode
15	Reserved		
	Format:	MBZ	
14	Force Sequential Access Enable		
	Format:	Enable	
	If ENABLED, the VERTEXDATA buffers are accessed sequentially, regardless of the value of 3DPRIMITIVE::VertexAccessType. The VertexID will still be equal to the index obtained from the Index Buffer if 3DPRIMITIVE::VertexAccessType is RANDOM. The 3DSTATE_VF_TOPOLOGY::Primitive Topology Type must be set to patchlist 1. If DISABLED, the VERTEXDATA buffers are accessed according to the value of 3DPRIMITIVE::VertexAccessType.		
	Programming Notes		
	Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.		
13	InstanceID Offset Enable		

3DSTATE_VF							
	<table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>If ENABLED, the InstanceID value optionally inserted into the vertex data, and used as an index for vertex element addressing when Instance Stride Enable is ENABLED, is offset by StartInstanceLocation. If DISABLED, InstanceID is not offset by StartInstanceLocation and instead is always 0-based.</p>	Format:	Enable				
Format:	Enable						
12	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
11	<p>VertexID Offset Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>If ENABLED, the VertexID value optionally inserted into the vertex data is offset by StartVertexLocation (SEQUENTIAL draws) or BaseVertexLocation (RANDOM draws). If DISABLED, VertexID is not offset by these values and instead is always 0-based</p>	Format:	Enable				
Format:	Enable						
10	<p>Sequential Draw Cut Index Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>If ENABLED, vertex indices in SEQUENTIAL 3DPRIMITIVE commands are compared to the Cut Index (specified below). When the vertex index matches the Cut Index any previous topology is terminated. If DISABLED, vertex indices are not compared to the Cut Index. This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details.</p> <table border="1"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.</td> </tr> </table>	Format:	Enable	Programming Notes		Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.	
Format:	Enable						
Programming Notes							
Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.							
9	<p>Component Packing Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>If ENABLED, vertex element component packing (as specified by 3DSTATE_VF_COMPONENT_PACKING) is performed before vertices are written into the URB. If DISABLED, no component packing is performed - all components of valid vertex elements will be stored in the URB.</p>	Format:	Enable				
Format:	Enable						
8	<p>Indexed Draw Cut Index Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>If ENABLED, vertex indices in RANDOM 3DPRIMITIVE commands are compared to the Cut Index (specified below). When the vertex index matches the Cut Index any previous topology is terminated. If DISABLED, vertex indices are not compared to the Cut Index and are used strictly as indices into vertex buffers. This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details.</p> <table border="1"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.</td> </tr> </table>	Format:	Enable	Programming Notes		Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.	
Format:	Enable						
Programming Notes							
Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.							

3DSTATE_VF						
	7:0	DWord Length <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>0h Excludes DWord (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	0h Excludes DWord (0,1)	Format:	=n
Default Value:	0h Excludes DWord (0,1)					
Format:	=n					
1	31:0	VF State Body <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>3DSTATE_VF_BODY</td> </tr> </table>	Format:	3DSTATE_VF_BODY		
Format:	3DSTATE_VF_BODY					

3DSTATE_VF_SGVS_2

3DSTATE_VF_SGVS_2		
Source:	RenderCS	
Length Bias:	2	
<p>This command is used to control the insertion of the Extended Parameter (XP0-2) System-Generated Values (SGVs) into an input Vertex URB Entry (VUE) (available as input to a VS thread). The insertions are individually-controlled. The insertion locations are specified as 128-bit element locations (starting at the beginning of the VUE) and 32-bit component within those specified elements. The SGV values can be inserted either (a) within a valid vertex element (in which case the value overwrites the value specified via 3DSTATE_VERTEX_ELEMENTS) or (b) beyond the last valid vertex element written to the URB. This permits some orthogonality between the programming of vertex elements (which typically is known at draw time) and programming of SGV insertion (which is associated with the shader). There are some restrictions however (see below). If an SGV is inserted beyond the last valid vertex element, zeroes are first inserted in the VUE after the last valid vertex element up to and including the vertex element receiving an SGV. If both of the SGVs are enabled for insertion, the zeroes will extend to the last vertex element receiving and SGV. Then the SGV(s) are inserted.</p> <p>The sources for these SGV values are derived from 3DPRIMITIVE command parameters. Controls in the 3DPRIMITIVE command determine whether (a) the parameters are directly defined via in-line command DWords, (b) the parameters are indirectly specified by command stream registers, or (c) the parameters are not included in the 3DPRIMITIVE command and therefore default to 0. Refer to the 3DPRIMITIVE command description for details on these different cases.</p> <p>The states included in this command are used to (a) enable/disable specific XP* SGV insertions and (b) for XP0 and XP1, specify which 3DPRIMITIVE parameters are used to source the inserted SGV value.</p> <p>The insertion of SGV values occurs before any component packing (3DSTATE_VF_COMPONENT_PACKING). Therefore the Element Offsets and Component Numbers specified in this command refer to the pre-packed data, following 3DSTATE_VERTEX_ELEMENT processing.</p>		
Programming Notes		
Programming Restrictions:		
<ul style="list-style-type: none"> It is INVALID to specify that more than one SGV is to be stored in the same element/component location within the VUE. The states programmed by this command overwrite the state programmed by any previous commands. I.e., a specific SGV (if enabled) can only be inserted in one component of a vertex. It is INVALID to insert an SGV value past the end of the VUE entry (as determined by VS URB Entry Allocation Size) or past the 33rd vertex element. Therefore the programming of VS URB Entry Allocation Size needs to comprehend any SGV insertion requirements. It is INVALID to use this command to overwrite any portion of a 64-bit vertex element component. 		
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.		
DWord	Bit	Description
0	31:29	Command Type Default Value: 3h GFXPIPE

3DSTATE_VF_SGVS_2		
		Format: OpCode
	28:27	Command SubType Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode Default Value: 56h 3DSTATE_VF_SGVS_2 Format: OpCode
	15:8	Reserved Format: MBZ
	7:0	DWord Length Default Value: 1h Excludes DWord (0,1) Format: =n
1..2	63:0	VF SGVS 2 State Body Format: 3DSTATE_VF_SGVS_2_BODY

3DSTATE_VF_SGVS

3DSTATE_VF_SGVS		
Source:	RenderCS	
Length Bias:	2	
<p>This command is used to control the insertion of the VertexID and InstanceID System-Generated Values (SGVs) into an input Vertex URB Entry (VUE) (available as input to a VS thread). VertexID and InstanceID insertion can be individually controlled. The insertion locations are specified as 128-bit element locations (starting at the beginning of the VUE) and the 32-bit component within those specified elements. The SGV values can be inserted either (a) within a valid vertex element (in which case the value overwrites the value specified via 3DSTATE_VERTEX_ELEMENTS) or (b) beyond the last valid vertex element written to the URB. This permits some orthogonality between the programming of vertex elements (which typically is known at draw time) and programming of SGV insertion (which is associated with the shader). There are some restrictions however (see below). If an SGV is inserted beyond the last valid vertex element, zeroes are first inserted in the VUE after the last valid vertex element up to and including the vertex element receiving an SGV. If both of the SGVs are enabled for insertion, the zeroes will extend to the last (largest index) vertex element receiving an SGV. Then the SGV(s) are inserted.</p> <p>The insertion of SGV values occurs before any component packing (3DSTATE_VF_COMPONENT_PACKING). Therefore the Element Offsets and Component Numbers specified in this command refer to the pre-packed data, following 3DSTATE_VERTEX_ELEMENT processing.</p>		
Programming Notes		
<p>Programming Restrictions:</p> <ul style="list-style-type: none"> It is INVALID to store both the VertexID and InstanceID in the same element/component location within the VUE. The states programmed by this command overwrite the state programmed by any previous commands. I.e., VertexID and InstanceID (if enabled) can only be inserted in one component of a vertex. It is INVALID to insert an SGV value past the end of the VUE entry (as determined by VS URB Entry Allocation Size) or past the 33rd vertex element. Therefore the programming of VS URB Entry Allocation Size needs to comprehend any SGV insertion requirements. It is INVALID to use this command to overwrite any portion of a 64-bit vertex element component. It is INVALID to use this command to overwrite a EdgeFlag vertex element component or any vertex element beyond it. 		
<p>When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
	Format: OpCode	
	28:27	Command SubType
Default Value: 3h GFXPIPE_3D		

3DSTATE_VF_SGVS		
		Format: OpCode
	26:24	3D Command Opcode Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode Default Value: 4Ah 3DSTATE_VF_SGVS Format: OpCode
	15:8	Reserved Format: MBZ
	7:0	DWord Length Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:0	VF SGVS State Body Format: 3DSTATE_VF_SGVS_BODY

3DSTATE_VF_STATISTICS

DWord		Bit	Description
Source:		RenderCS	
Length Bias:		1	
<p>The VF stage tracks two pipeline statistics, the number of vertices fetched and the number of objects generated. VF will increment the appropriate counter for each when statistics gathering is enabled by issuing the 3DSTATE_VF_STATISTICS command with the [Statistics Enable] bit set.</p>			
Programming Notes			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.			
0	31:29	Command Type Default Value: 3h GFXPIPE Format: Opcode	
	28:27	Command SubType Default Value: 1h GFXPIPE_SINGLE_DW Format: Opcode	
	26:24	3D Command Opcode Default Value: 0h 3DSTATE_PIPELINED Format: Opcode GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)	
	23:16	3D Command Sub Opcode Default Value: 0Bh 3DSTATE_VF_STATISTICS Format: Opcode GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)	
	15:1	Reserved Format: MBZ	
	0	Statistics Enable Format: Enable If ENABLED, VF will increment the pipeline statistics counters IA_VERTICES_COUNT and IA_PRIMITIVES_COUNT for each vertex fetched and each object output, respectively, for 3DPRIMITIVE commands issued subsequently. If DISABLED, these counters will not be incremented for subsequent 3DPRIMITIVE commands.	
Programming Notes			
When a 3DPRIMITIVE command with POSH Enable set is executed from the RCS command stream, VF statistics gathering is inhibited for that command.			

3DSTATE_VF_TOPOLOGY

3DSTATE_VF_TOPOLOGY			
Source:	RenderCS		
Length Bias:	2		
This command specifies the VF stage's Topology state which can be used to override the Primitive Topology Type in subsequent 3DPRIMITIVE commands.			
Programming Notes			
When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline, and only the following topologies are valid: 3DPRIM_TRILIST, 3DPRIM_TRISTRIP, 3DPRIM_TRISTRIP_REV.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
Default Value:		0h 3DSTATE_PIPELINED	
Format:		OpCode	
23:16	3D Command Sub Opcode		
	Default Value:	4Bh 3DSTATE_VF_TOPOLOGY	
	Format:	OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	
	Format:	=n	
1	31:0	VF Topology State Body	
		Format:	3DSTATE_VF_TOPOLOGY_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_CC

3DSTATE_VIEWPORT_STATE_POINTERS_CC			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_VIEWPORT_STATE_POINTERS_CC command is used to define the location of fixed functions' viewport state table.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
23:16	3D Command Sub Opcode		
	Default Value:	23h 3DSTATE_VIEWPORT_STATE_POINTERS_CC	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
1	31:0	Viewport State Pointers CC State Body	
		Format:	3DSTATE_VIEWPORT_STATE_POINTERS_CC_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP			
Source:	RenderCS		
Length Bias:	2		
The 3DSTATE_VIEWPORT_STATE_POINTERS_CLIP command is used to define the location of fixed functions' viewport state table.			
Restriction			
When executed in POCS command stream, this programs viewport state of the POCS pipeline.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		21h 3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h DWORD_COUNT_n	
	Format:	=n	
1	31:0	Viewport State Pointers SF Clip State Body	
		Format:	3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP_BODY

3DSTATE_VS

3DSTATE_VS		
Source:	RenderCS, PositionCS	
Length Bias:	2	
This command specifies most of the state used by the Vertex Shader (VS) stage.		
Programming Notes		
When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline. When executed in the RCS command stream, this command programs state for the VS stage of the RCS pipeline.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Command SubType
		Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	3D Command Opcode
		Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	3D Command Sub Opcode
		Default Value: 10h 3DSTATE_VS Format: OpCode
15	Reserved	
14:8	Reserved	
7:0	DWord Length	
	Default Value: 7h Excludes DWord (0,1) Format: =n	
1..8	255:0	VS State Body
		Format: 3DSTATE_VS_BODY

3DSTATE_WM_CHROMAKEY

3DSTATE_WM_CHROMAKEY			
Source:		RenderCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		4Ch 3DSTATE_WM_CHROMAKEY	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	Dword Length		
	Default Value:	0h Excludes Dword (0,1)	
	Format:	=n	
	Total Length - 2		
1	31:0	WM Chromakey State Body	
		Format:	3DSTATE_WM_CHROMAKEY_BODY

3DSTATE_WM_DEPTH_STENCIL

3DSTATE_WM_DEPTH_STENCIL			
Source:	RenderCS		
Length Bias:	2		
This command replaces the indirect state DEPTH_STENCIL_STATE with an inline state command.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	4Eh 3DSTATE_WM_DEPTH_STENCIL
		Format:	OpCode
15:13	Reserved		
	Format:	MBZ	
12	Depth State Modify Disable		
	Format:	Disable	
When this bit is set, the following fields will be ignored:			
<ul style="list-style-type: none"> Depth Test Function Depth Test Enable Depth Buffer Write Enable 			
11	Stencil State Modify Disable		
	Format:	Disable	
When this bit is set, the following fields will be ignored:			
<ul style="list-style-type: none"> Stencil Fail Op Stencil Pass Depth Fail Op Stencil Pass Depth Pass Op Backface Stencil Test Function Backface Stencil Fail Op Backface Stencil Pass Depth Fail Op 			

3DSTATE_WM_DEPTH_STENCIL						
		<ul style="list-style-type: none"> • Backface Stencil Pass Depth Pass Op • Stencil Test Function • Double Sided Stencil Enable • Stencil Test Enable • Stencil Buffer Write Enable 				
	10	<p>Stencil Write Mask Modify Disable</p> <table border="1"> <tr> <td>Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Stencil Write Mask • Backface Stencil Write Mask 	Format:	Disable		
Format:	Disable					
	9	<p>Stencil Test Mask Modify Disable</p> <table border="1"> <tr> <td>Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Stencil Test Mask • Backface Stencil Test Mask 	Format:	Disable		
Format:	Disable					
	8	<p>Stencil Reference Value Modify Disable</p> <table border="1"> <tr> <td>Format:</td> <td>Disable</td> </tr> </table> <p>When this bit is set, the following fields will be ignored:</p> <ul style="list-style-type: none"> • Stencil Reference Value • Backface Stencil Reference Value 	Format:	Disable		
Format:	Disable					
	7:0	<p>Dword Length</p> <table border="1"> <tr> <td>Default Value:</td> <td>02h Excludes Dword (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>Total Length - 2</p>	Default Value:	02h Excludes Dword (0,1)	Format:	=n
Default Value:	02h Excludes Dword (0,1)					
Format:	=n					
1..3	95:0	<p>WM Depth Stencil State Body</p> <table border="1"> <tr> <td>Format:</td> <td>3DSTATE_WM_DEPTH_STENCIL_BODY</td> </tr> </table>	Format:	3DSTATE_WM_DEPTH_STENCIL_BODY		
Format:	3DSTATE_WM_DEPTH_STENCIL_BODY					

3DSTATE_WM_HZ_OP

3DSTATE_WM_HZ_OP			
Source:	RenderCS		
Length Bias:	2		
This command provides for clearing Z and/or stencil or resolving either HZ buffer or Z buffer.			
Programming Notes			
As this command generates an implicit rectangle, SW must make sure any MMIO register writes following WM_HZ_OP must be preceded by PIPE_CONTROL with Command Streamer Stall Enable bit set.			
<p>3DSTATE_DRAWING_RECTANGLE must be programmed such that it does not clip the HZ_OP command's rectangle. See programming notes for X/Y Min and X/Y Max below.</p> <p>3DSTATE_DRAWING_RECTANGLE command must come before 3DSTATE_WM_HZ_OP in the command buffer</p> <p>Caution: There is a difference in how X/Y coordinates are interpreted by 3DSTATE_DRAWING_RECTANGLE vs. 3DSTATE_WM_HZ_OP.</p> <p>HZ_OP rectangle parameters are exclusive on max side, for example to have 8x4 rectangle we would program X Min = 0, Y Min = 0, X Max = 8, Y Max = 4.</p> <p>Draw Rectangle parameters are inclusive on max side, meaning, for 8x4 rectangle the values would be X Min = 0, Y Min = 0, X Max = 7, Y Max = 3.</p>			
3DSTATE_MULTISAMPLE packet must be used prior to this packet to change the Number of Multisamples. This packet must not be used to change Number of Multisamples in a rendering sequence.			
3DSTATE_RASTER if used must be programmed prior to using this packet.			
SW should flush the Depth Cache AFTER the following WM_HZ_OP commands			
<ol style="list-style-type: none"> 1. Depth Clear 2. HIZ Resolve 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
		Default Value:	52h 3DSTATE_WM_HZ_OP
		Format:	OpCode
	15:9	Reserved	
		Format:	MBZ

3DSTATE_WM_HZ_OP			
	8	Reserved Format: MBZ	
	7:0	Dword Length Format: =n	
		Total Length - 2	
		Value	Name
		03h Excludes Dword (0,1) [Default]	
1..4	127:0	WM HZ OP State Body Format: 3DSTATE_WM_HZ_OP_BODY	

3DSTATE_WM

3DSTATE_WM			
Source:		RenderCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	0h 3DSTATE_PIPELINED
		Format:	OpCode
	23:16	3D Command Sub Opcode	
Default Value:		14h 3DSTATE_WM	
Format:		OpCode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1	31:0	WM State Body	
		Format:	3DSTATE_WM_BODY

A64 Byte Scattered Read MSD

MSD1R_A64_BS - A64 Byte Scattered Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: 10h Format: Opcode A64 Scattered Read message
	13	Invalidate After Read Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	SIMD Mode Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	Data Elements Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	A64 Scattered Message Subtype

MSD1R_A64_BS - A64 Byte Scattered Read MSD		
	Default Value:	0h
	Format:	Opcode
	Byte Read/Write subtype	
7:0	Binding Table Index	
	Format:	MDC_STATELESS
Specifies the message is stateless		

A64 Byte Scattered Write MSD

MSD1W_A64_BS - A64 Byte Scattered Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: _____ 1Ah Format: _____ Opcode A64 Scattered Write message
	13	Reserved Format: _____ MBZ
	12	SIMD Mode Format: _____ MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	Data Elements Format: _____ MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	A64 Scattered Message Subtype Default Value: _____ 0h Format: _____ Opcode

MSD1W_A64_BS - A64 Byte Scattered Write MSD			
	Byte Read/Write subtype		
7:0	Binding Table Index <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_STATELESS</td> </tr> </table> Specifies the message is stateless	Format:	MDC_STATELESS
Format:	MDC_STATELESS		

A64 Dword Scattered Read MSD

MSD1R_A64_DWS - A64 Dword Scattered Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: _____ 10h Format: _____ Opcode A64 Scattered Read message
	13	Invalidate After Read Format: _____ MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	SIMD Mode Format: _____ MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	Data Elements Format: _____ MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	A64 Scattered Message Subtype

MSD1R_A64_DWS - A64 Dword Scattered Read MSD		
	Default Value:	1h
	Format:	Opcode
	Dword Read/Write subtype	
7:0	Binding Table Index	
	Format:	MDC_STATELESS
Specifies the message is stateless		

A64 Dword Scattered Write MSD

MSD1W_A64_DWS - A64 Dword Scattered Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	Reserved Format: MBZ
	12	SIMD Mode Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	Data Elements Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	A64 Scattered Message Subtype Default Value: 1h Format: Opcode

MSD1W_A64_DWS - A64 Dword Scattered Write MSD			
	Dword Read/Write subtype		
7:0	Binding Table Index <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_STATELESS</td> </tr> </table> Specifies the message is stateless	Format:	MDC_STATELESS
Format:	MDC_STATELESS		

A64 Dword Untyped Atomic Float with Return Data Operation MSD

MSD1R_A64_DWAF - A64 Dword Untyped Atomic Float with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved
		Format: MBZ
	28:25	Message Length
		Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length
		Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present
		Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type
		Default Value: 1Dh
Format: Opcode A64 Untyped Atomic Float Operation message		
13	Return Data Control	
	Default Value: 1h	
	Format: Opcode Specifies that return data is sent back to the thread.	
12	SIMD Mode	
	Format: MDC_SM2S Only SIMD8 operations are supported.	
11	Data Width	
	Default Value: 0h	

MSD1R_A64_DWAF - A64 Dword Untyped Atomic Float with Return Data Operation MSD

		Format:	Opcode
		Operations are on 32-bit floats.	
10:8	Atomic Float Operation	Format:	MDC_FOP
		Specifies the atomic float operation to be performed.	
7:0	Binding Table Index	Format:	MDC_STATELESS
		Specifies the message is stateless	

A64 Dword Untyped Atomic Float Write Only Operation MSD

MSD1W_A64_DWAF - A64 Dword Untyped Atomic Float Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: 1Dh Format: Opcode A64 Untyped Atomic Float Operation message
	13	Return Data Control Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	SIMD Mode Format: MDC_SM2S Description Only SIMD8 operations are supported.
	11	Data Width

MSD1W_A64_DWAF - A64 Dword Untyped Atomic Float Write Only Operation MSD

	Default Value:	0h
	Format:	Opcode
	Operations are on 32-bit floats.	
10:8	Atomic Float Operation Type	
	Format:	MDC_FOP
	Specifies the atomic float operation to be performed.	
7:0	Binding Table Index	
	Format:	MDC_STATELESS
	Specifies the message is stateless	

A64 Dword Untyped Atomic Integer with Return Data Operation MSD

MSD1R_A64_DWAI - A64 Dword Untyped Atomic Integer with Return Data Operation MSD						
Source:	EuSubFunctionDataPort1					
Length Bias:	1					
DWord	Bit	Description				
0	31:29	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ				
	28:25	Message Length <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4		
	Format:	U4				
	24:20	Response Length <table border="1"> <tr> <td>Format:</td> <td>U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5		
	Format:	U5				
	19	Header Present <table border="1"> <tr> <td>Format:</td> <td>MDC_MHF</td> </tr> </table> <p>The message forbids a header</p>	Format:	MDC_MHF		
	Format:	MDC_MHF				
	18:14	Message Type <table border="1"> <tr> <td>Default Value:</td> <td>12h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table> <p>A64 Untyped Atomic Integer Operation message</p>	Default Value:	12h	Format:	Opcode
	Default Value:	12h				
Format:	Opcode					
13	Return Data Control <table border="1"> <tr> <td>Default Value:</td> <td>1h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table> <p>Specifies that return data is sent back to the thread.</p>	Default Value:	1h	Format:	Opcode	
Default Value:	1h					
Format:	Opcode					
12	Data Width <table border="1"> <tr> <td>Default Value:</td> <td>0h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table> <p>Operations are on 32-bit integers</p>	Default Value:	0h	Format:	Opcode	
Default Value:	0h					
Format:	Opcode					

MSD1R_A64_DWAI - A64 Dword Untyped Atomic Integer with Return Data Operation MSD

	11:8	Atomic Integer Operation <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MDC_AOP</td> </tr> </table> <p>Specifies the atomic integer operation to be performed.</p>	Format:	MDC_AOP
	Format:	MDC_AOP		
7:0	Binding Table Index <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MDC_STATELESS</td> </tr> </table> <p>Specifies the message is stateless</p>	Format:	MDC_STATELESS	
Format:	MDC_STATELESS			

A64 Dword Untyped Atomic Integer Write Only Operation MSD

MSD1W_A64_DWAI - A64 Dword Untyped Atomic Integer Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF The message forbids a header
	18:14	Message Type Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
	13	Return Data Control Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	Data Width Default Value: 0h Format: Opcode Operations are on 32-bit integers
	11:8	Atomic Integer Operation

MSD1W_A64_DWAI - A64 Dword Untyped Atomic Integer Write Only Operation MSD

		Format:	MDC_AOP
		Specifies the atomic integer operation to be performed.	
	7:0	Binding Table Index	
		Format:	MDC_STATELESS
		Specifies the message is stateless	

A64 Hword Block Read MSD

MSD1R_A64_HWB - A64 Hword Block Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	Invalidate After Read Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	A64 Block Message Subtype Default Value: 3h Format: Opcode Hword Block Read/Write subtype
	10:8	Data Elements Format: MDC_A64_DB_HW Specifies the number of contiguous Hwords to be read or written

MSD1R_A64_HWB - A64 Hword Block Read MSD

	7:0	Binding Table Index	
		Format:	MDC_STATELESS
		Specifies the message is stateless	

A64 Hword Block Write MSD

MSD1W_A64_HWB - A64 Hword Block Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: _____ 15h Format: _____ Opcode A64 Hword Block Write message
	13	Reserved Format: _____ MBZ
	12:11	A64 Block Message Subtype Default Value: _____ 3h Format: _____ Opcode Hword Block Read/Write subtype
	10:8	Data Elements Format: _____ MDC_A64_DB_HW Specifies the number of contiguous Hwords to be read or written
	7:0	Binding Table Index Format: _____ MDC_STATELESS

MSD1W_A64_HWB - A64 Hword Block Write MSD		
		Specifies the message is stateless

A64 Oword Aligned Block Read MSD

MSD1R_A64_OWAB - A64 Oword Aligned Block Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	Reserved Format: MBZ
	12:11	A64 Block Message Subtype Default Value: 1h Format: Opcode Oword Aligned Block Read subtype
	10:8	Data Elements Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be read
	7:0	Binding Table Index Format: MDC_STATELESS Specifies the message is stateless

A64 Oword Aligned Block Write MSD

MSD1W_A64_OWAB - A64 Oword Aligned Block Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: 15h Format: Opcode A64 Oword Block Write message
	13	Reserved Format: MBZ
	12:11	A64 Block Message Subtype Default Value: 1h Format: Opcode Oword Aligned Block Write subtype
	10:8	Data Elements Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be written
	7:0	Binding Table Index Format: MDC_STATELESS Specifies the message is stateless

A64 Oword Block Read MSD

MSD1R_A64_OWB - A64 Oword Block Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	Invalidate After Read Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	A64 Block Message Subtype Default Value: 0h Format: Opcode Oword Block Read/Write subtype
	10:8	Data Elements Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be read or written

MSD1R_A64_OWB - A64 Oword Block Read MSD			
7:0	<p>Binding Table Index</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_STATELESS</td> </tr> </table> <p>Specifies the message is stateless</p>	Format:	MDC_STATELESS
Format:	MDC_STATELESS		

A64 Oword Block Write MSD

MSD1W_A64_OWB - A64 Oword Block Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: 15h Format: Opcode A64 Oword Block Write message
	13	Reserved Format: MBZ
	12:11	A64 Block Message Subtype Default Value: 0h Format: Opcode Oword Block Read/Write subtype
	10:8	Data Elements Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	Binding Table Index Format: MDC_STATELESS

MSD1W_A64_OWB - A64 Oword Block Write MSD		
		Specifies the message is stateless

A64 Qword Scattered Read MSD

MSD1R_A64_QWS - A64 Qword Scattered Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: 10h Format: Opcode A64 Scattered Read message
	13	Invalidate After Read Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	SIMD Mode Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	Data Elements Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	A64 Scattered Message Subtype

MSD1R_A64_QWS - A64 Qword Scattered Read MSD		
	Default Value:	2h
	Format:	Opcode
	Qword Read/Write subtype	
	Binding Table Index	
7:0	Format:	MDC_STATELESS
Specifies the message is stateless		

A64 Qword Scattered Write MSD

MSD1W_A64_QWS - A64 Qword Scattered Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	Reserved Format: MBZ
	12	SIMD Mode Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	Data Elements Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	A64 Scattered Message Subtype Default Value: 2h Format: Opcode

MSD1W_A64_QWS - A64 Qword Scattered Write MSD			
	Qword Read/Write subtype		
7:0	Binding Table Index <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_STATELESS</td> </tr> </table> Specifies the message is stateless	Format:	MDC_STATELESS
Format:	MDC_STATELESS		

A64 Qword Untyped Atomic Integer with Return Data Operation MSD

MSD1R_A64_QWAI - A64 Qword Untyped Atomic Integer with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF The message forbids a header
	18:14	Message Type Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
		Return Data Control Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	Data Width Default Value: 1h Format: Opcode Operations are on 64-bit integers

MSD1R_A64_QWAI - A64 Qword Untyped Atomic Integer with Return Data Operation MSD

	11:8	Atomic Integer Operation <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_AOP</td> </tr> </table> <p>Specifies the atomic integer operation to be performed.</p>	Format:	MDC_AOP
	Format:	MDC_AOP		
7:0	Binding Table Index <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_STATELESS</td> </tr> </table> <p>Specifies the message is stateless</p>	Format:	MDC_STATELESS	
Format:	MDC_STATELESS			

A64 Qword Untyped Atomic Integer Write Only Operation MSD

MSD1W_A64_QWAI - A64 Qword Untyped Atomic Integer Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF The message forbids a header
	18:14	Message Type Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
	13	Return Data Control Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	Data Width Default Value: 1h Format: Opcode Operations are on 64-bit integers
	11:8	Atomic Integer Operation

MSD1W_A64_QWAI - A64 Qword Untyped Atomic Integer Write Only Operation MSD

		Format:	MDC_AOP
		Specifies the atomic integer operation to be performed.	
	7:0	Binding Table Index	
		Format:	MDC_STATELESS
		Specifies the message is stateless	

A64 Untyped Surface Read MSD

MSD1R_A64_US - A64 Untyped Surface Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: _____ 11h Format: _____ Opcode A64 Untyped Surface Read message
	13:12	SIMD Mode Format: _____ MDC_SM3 Specifies the SIMD mode of the message (number of slots processed)
	11:8	Channel Mask Format: _____ MDC_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	Binding Table Index Format: _____ MDC_STATELESS Specifies the message is stateless

A64 Untyped Surface Write MSD

MSD1W_A64_US - A64 Untyped Surface Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: 19h Format: Opcode A64 Untyped Surface Write message
	13:12	SIMD Mode Format: MDC_SM3 Specifies the SIMD mode of the message (number of slots processed)
	11:8	Channel Mask Format: MDC_UW_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	Binding Table Index Format: MDC_STATELESS Specifies the message is stateless

A64 Word Untyped Atomic Float with Return Data Operation MSD

MSD1R_A64_WAF - A64 Word Untyped Atomic Float with Return Data Operation MSD		
Source:	EuSubFunctionDataPort1	
Length Bias:	1	
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type Default Value: 1Eh Format: Opcode A64 Untyped Atomic Half Float Operation message
	13	Return Data Control Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	Reserved Format: MBZ
	11	Data Width Default Value: 0h Format: Opcode Operations are on 32-bit floats.

MSD1R_A64_WAF - A64 Word Untyped Atomic Float with Return Data Operation MSD			
	<p>10:8 Atomic Float Operation</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_FOP</td> </tr> </table> <p>Specifies the atomic float operation to be performed.</p>	Format:	MDC_FOP
	Format:	MDC_FOP	
<p>7:0 Binding Table Index</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_STATELESS</td> </tr> </table> <p>Specifies the message is stateless</p>	Format:	MDC_STATELESS	
Format:	MDC_STATELESS		

A64 Word Untyped Atomic Float Write Only Operation MSD

MSD1W_A64_WAF - A64 Word Untyped Atomic Float Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved
		Format: MBZ
	28:25	Message Length
		Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length
		Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present
		Format: MDC_MHF Indicates that the message forbids a header
	18:14	Message Type
		Default Value: 1Eh
Format: Opcode A64 Untyped Atomic Half Float Operation message		
13	Return Data Control	
	Default Value: 0h	
	Format: Opcode Specifies that no return data is sent back to the thread.	
12	Reserved	
	Format: MBZ	
11	Data Width	
	Default Value: 0h	
	Format: Opcode Operations are on 32-bit floats.	

MSD1W_A64_WAF - A64 Word Untyped Atomic Float Write Only Operation MSD	
	10:8 Atomic Float Operation Type Format: MDC_FOP Specifies the atomic float operation to be performed.
	7:0 Binding Table Index Format: MDC_STATELESS Specifies the message is stateless

A64 Word Untyped Atomic Integer with Return Data Operation MSD

MSD1R_A64_WAI - A64 Word Untyped Atomic Integer with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved
		Format: MBZ
	28:25	Message Length
		Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length
		Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present
		Format: MDC_MHF The message forbids a header
	18:14	Message Type
		Default Value: 13h
Format: Opcode A64 Untyped Atomic Half Integer Operation message		
13	Return Data Control	
	Default Value: 1h	
	Format: Opcode Specifies that return data is sent back to the thread.	
12	Data Width	
	Default Value: 0h	
	Format: Opcode Operations are on 16-bit integers	

MSD1R_A64_WAI - A64 Word Untyped Atomic Integer with Return Data Operation MSD

	11:8	Atomic Integer Operation	
		Format:	MDC_AOP
		Specifies the atomic integer operation to be performed.	
	7:0	Binding Table Index	
		Format:	MDC_STATELESS
		Specifies the message is stateless	

A64 Word Untyped Atomic Integer Write Only Operation MSD

MSD1W_A64_WAI - A64 Word Untyped Atomic Integer Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF The message forbids a header
	18:14	Message Type Default Value: 13h Format: Opcode A64 Untyped Atomic Half Integer Operation message
	13	Return Data Control Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	Data Width Default Value: 0h Format: Opcode Operations are on 16-bit integers
	11:8	Atomic Integer Operation

MSD1W_A64_WAI - A64 Word Untyped Atomic Integer Write Only Operation MSD

		Format:	MDC_AOP
		Specifies the atomic integer operation to be performed.	
	7:0	Binding Table Index	
		Format:	MDC_STATELESS
		Specifies the message is stateless	

Addition

add - Addition			
Source:	Eulsa		
Length Bias:	4		
The add instruction performs component-wise addition of src0 and src1 and stores the results in dst. Addition of two floating-point numbers follows rules in add (IEEE mode) or add (ALT mode).			
Format: [(pred)] add[.cmod] (exec_size) dst src0 src1			
Programming Notes			
Use a source modifier with add to implement subtraction.			
Syntax			
[(pred)] add[.cmod] (exec_size) reg reg reg [(pred)] add[.cmod] (exec_size) reg reg imm32			
Pseudocode			
<pre>Evaluate (WrEn); { for (n = 0; n < exec_size; n++) { dst.chan[n] = src0.chan[n] + src1.chan[n]; } if (WrEn.chan[n]) }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types		Dst Types	
*B,*W,*D		*B,*W,*D	
*B,*W,*D		F	
F		F	
HF		HF	
*B,*W,*D		HF	
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
	127:96	Src1.ImmValue[31:0]	
Exists If:		([Src1.IsImm]==true)	
125:122	Reserved		
	Exists If:	([Src1.IsImm]==false)	
	Format:	MBZ	
121:120	Src1.Mod		

add - Addition

	Exists If:	([Src1.IsImm]==false)	
	Format:	SrcMod	
119:116	Src1.VertStride		
	Exists If:	([Src1.IsImm]==false)	
	Format:	VertStride	
115:113	Src1.Width		
	Exists If:	([Src1.IsImm]==false)	
	Format:	Width	
112	Src1.AddrMode		
	Exists If:	([Src1.IsImm]==false)	
	Format:	AddrMode	
111:98	Src1.Operand		
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)	
	Format:	DirectOperand	
111:98	Src1.Operand		
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)	
	Format:	IndirectOperand	
97:96	Src1.HorzStride		
	Exists If:	([Src1.IsImm]==false)	
	Format:	HorzStride	
95:92	CondCtrl		
	Format:	FlagModifier	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==false)	
	Format:	RegDataType	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==true)	
	Format:	ImmDataType	
87:84	Src0.VertStride		
	Format:	VertStride	
83:81	Src0.Width		
	Format:	Width	
80	Src0.AddrMode		
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	([Src0.AddrMode]==Direct)	

add - Addition		
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	((Src0.AddrMode) == Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	((Dst.AddrMode) == Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	((Dst.AddrMode) == Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
1	true	
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
1	true	
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode

add - Addition

34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
		Description
	0	Normal [Default]
		Normal. Per channel write enable used for final write enable generation.
	1	NoMask
		NoMask. Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name
		Description
	0	NoCompaction [Default]
		No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted
		Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name
		Description
	0	Positive [Default]
		Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative
		Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl	

add - Addition			
	<table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl
Format:	PredCtrl		
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>	Format:	Header
Format:	Header		

Addition with Carry

addc - Addition with Carry			
Source:	Eulsa		
Length Bias:	4		
<p>The addc instruction performs component-wise addition of src0 and src1 and stores the results in dst; it also stores the carry into acc. If the operation produces a carry out, 0x00000001 is stored in acc, else 0x00000000 is stored in acc.</p>			
Format:	[(pred)] addc[.cmod] (exec_size) dst src0 src1		
Restriction			
AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.			
Syntax			
[(pred)] addc[.cmod] (exec_size) reg reg reg [(pred)] addc[.cmod] (exec_size) reg reg imm32			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] + src1.chan[n]; acc.chan[n] = carry(src0.chan[n] + src1.chan[n]); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N
Src Types	Dst Types		
UD	UD		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
	125:122	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
121:120	Src1.Mod		
	Exists If:	([Src1.IsImm]==false)	
Format:	SrcMod		

addc - Addition with Carry

	119:116	Src1.VertStride	
		Exists If:	([Src1.IsImm]==false)
		Format:	VertStride
	115:113	Src1.Width	
		Exists If:	([Src1.IsImm]==false)
		Format:	Width
	112	Src1.AddrMode	
		Exists If:	([Src1.IsImm]==false)
		Format:	AddrMode
	111:98	Src1.Operand	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
		Format:	DirectOperand
	111:98	Src1.Operand	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
		Format:	IndirectOperand
	97:96	Src1.HorzStride	
	Exists If:	([Src1.IsImm]==false)	
	Format:	HorzStride	
95:92	CondCtrl		
	Format:	FlagModifier	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==false)	
	Format:	RegDataType	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==true)	
	Format:	ImmDataType	
87:84	Src0.VertStride		
	Format:	VertStride	
83:81	Src0.Width		
	Format:	Width	
80	Src0.AddrMode		
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		

addc - Addition with Carry

		Exists If:	([Src0.AddrMode] == Indirect)
		Format:	IndirectOperand
65:64	Src0.HorzStride		
		Format:	HorzStride
63:50	Dst.Operand		
		Exists If:	((Dst.AddrMode) == Direct)
		Format:	DirectOperand
63:50	Dst.Operand		
		Exists If:	((Dst.AddrMode) == Indirect)
		Format:	IndirectOperand
49:48	Dst.HorzStride		
		Format:	HorzStride
47	Src1.IsImm		
	This field indicate that Source 1 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
		Format:	SrcMod
43:40	Src0.DataType		
		Exists If:	((Src0.IsImm) == false)
		Format:	RegDataType
43:40	Src0.DataType		
		Exists If:	((Src0.IsImm) == true)
		Format:	ImmDataType
39:36	Dst.DataType		
		Format:	RegDataType
35	Dst.AddrMode		
		Format:	AddrMode
34	Saturate		
		Format:	Saturate

addc - Addition with Carry

33	AccWrCtrl Format: AccWrCtrl	
32	AtomicCtrl Format: AtomicCtrl	
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
	Description	
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask. Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name
	Description	
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name
	Description	
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the	

addc - Addition with Carry			
	content of the selected flag register.		
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Arithmetic Shift Right

asr - Arithmetic Shift Right			
Source:	Eulsa		
Length Bias:	4		
<p>Perform component-wise arithmetic right shift of the bits in src0 by the shift count indicated in src1, storing the results in dst. If src0 has a signed type, insert copies of src0's sign bit in the number of MSBs indicated by the shift count. Otherwise insert 0 bits. The shift count is taken from the low five bits of src1, regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type. For positive values, this operation is $\text{src0} / 2^{\text{shiftCount}}$ and for negative values, this operation is $\text{src0} / 2^{\text{shiftCount} - 1}$.</p>			
Format:	[(pred)] asr[.cmo] (exec_size) dst src0 src1		
Programming Notes			
If src0 is -1, the result is -1 regardless of the shift count.			
For unsigned src0 types, asr and shr produce the same result.			
Syntax			
[(pred)] asr[.cmo] (exec_size) reg reg reg [(pred)] asr[.cmo] (exec_size) reg reg imm32			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { shiftCnt = src1.chan[n] & 0x1F; // Always use low 5 bits for shift count. shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] & 0x1F if (src0.chan[n] >= 0) { dst.chan[n] = src0.chan[n] » shiftCnt; } else { int maskLSB = pow(2, shiftCnt) - 1; if (maskLSB & src0.chan[n] == 0) { dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt); } else { dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt) - 1; } } } } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)

asr - Arithmetic Shift Right

	Format:	MBZ
127:96	Src1.ImmValue[31:0]	
	Exists If:	([Src1.IsImm]==true)
125:122	Reserved	
	Exists If:	([Src1.IsImm]==false)
	Format:	MBZ
121:120	Src1.Mod	
	Exists If:	([Src1.IsImm]==false)
	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	([Src1.IsImm]==false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	([Src1.IsImm]==false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	

asr - Arithmetic Shift Right

	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	(([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	(([Src0.AddrMode]==Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	(([Dst.AddrMode]==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	(([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	(([Src0.IsImm]==false)
	Format:	RegDataType

asr - Arithmetic Shift Right

43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==true)	
	Format:	ImmDataType	
39:36	Dst.DataType		
	Format:	RegDataType	
35	Dst.AddrMode		
	Format:	AddrMode	
34	Saturate		
	Format:	Saturate	
33	AccWrCtrl		
	Format:	AccWrCtrl	
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:	MBZ	
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no		

asr - Arithmetic Shift Right

	<p>predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		Format:	PredCtrl							
Format:	PredCtrl										
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>										
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>										
21:19	<p>ChanOff</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. This can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>		Format:	ChanOff							
Format:	ChanOff										
18:16	<p>ExecSize</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>		Format:	ExecSize							
Format:	ExecSize										
15:0	<p>Header</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>		Format:	Header							
Format:	Header										

Average

avg - Average			
Source:	Eulsa		
Length Bias:	4		
<p>The avg instruction performs component-wise integer average of src0 and src1 and stores the results in dst. An integer average uses integer upward rounding. It is equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic right shift to this intermediate value.</p>			
<p>Format:</p> <p>The avg instruction performs component-wise integer average of src0 and src1 and stores the results in dst. An integer average uses integer upward rounding. It is equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic right shift to this intermediate value.</p>			
Syntax			
<pre>[(pred)] avg[.cmod] (exec_size) reg reg reg [(pred)] avg[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = (src0.chan[n] + src1.chan[n] + 1) » 1; // Use arithmetic shift right. } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
125:122	Reserved		
	Exists If:	([Src1.IsImm]==false)	
Format:	MBZ		
121:120	Src1.Mod		
	Exists If:	([Src1.IsImm]==false)	
Format:	SrcMod		
119:116	Src1.VertStride		

avg - Average		
	Exists If:	([Src1.IsImm]==false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	([Src1.IsImm]==false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	([Src1.IsImm]==false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)

avg - Average

	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	((Dst.AddrMode)==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	((Dst.AddrMode)==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm)==false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm)==true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	

avg - Average		
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
	Description	
	0	Normal [Default]
		Normal. Per channel write enable used for final write enable generation.
	1	NoMask
		NoMask. Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name
	Description	
	0	NoCompaction [Default]
		No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted
		Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name
	Description	
	0	Positive [Default]
		Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative
		Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl	
	Format:	PredCtrl
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.	

avg - Average			
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

AVP_BSD_OBJECT

AVP_BSD_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>The AVP Pipeline is selected with the Media Instruction Opcode "8h" for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The AVP_BSD_OBJECT command sends to HW a tile at a time from an AV1 bitstream, starting with the first coded byte of the tile, not including the prefixed tile byte size. The bit stream of a tile, tile group, and of a frame may end with trailing bits and extra padding zero bytes. The prefixed tile byte size includes all the trailing bits and padding zero bytes at the end of a tile.</p> <p>Each tile's coded/compressed bitstream is started and ended at a byte boundary.</p> <p>HW is not required to parse the trailing bits and padding zero bytes. HW can stop processing right after it has completed the decoding of the last block in the tile. Potentially, error checking can be implemented to detect the trailing bits and padding zeros, but is not implemented in this generation of AVP Pipeline.</p> <p>here can be multiple tiles in an AV1 frame and thus this command can be issued multiple times per frame. A coded frame minimum has at least 1 tile definition, i.e a tile can cover the entire frame, unless the frame size exceeds the max allowed tile size limits in pixels, then the frame must contain more than 1 tile. There is no compressed header in AV1, hence AVP_BSD_OBJECT command is only used to process the bitstream of each individual tile of a frame.</p> <p>The AVP_BSD_OBJECT command must be the last command issued in the sequence of batch commands before the AVP Pipeline starts decoding. Prior to issuing this command, it is assumed that all configuration parameters needed by the AVP Pipeline have been loaded in a specific order, including workload configuration registers and configuration tables. When this command is issued, the AVP Pipeline is waiting for bitstream data to be presented to its bitstream input shift register.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = AV1 = 3h	
	22:16	Media Instruction Command	
		Default Value:	20h AVP_BSD_OBJECT_STATE
		Format:	OpCode
	15:12	Reserved	

AVP_BSD_OBJECT										
		<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									
	11:0	<p>Dword Length</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>=n</td> </tr> </table> <p>(Excludes Dwords 0, 1).</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td>1h</td> <td></td> </tr> </tbody> </table>	Format:	=n	Value	Name	1h			
Format:	=n									
Value	Name									
1h										
1	31:0	<p>Tile Indirect BSD Data Length</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U32</td> </tr> </table> <p>It specifies the compressed bitstream byte size of a tile. Each tile is started and ended at a byte boundary. It has the same value as the prefixed tile byte size read from the bitstream at the beginning of a tile. The Data Length does not include this prefixed tile byte size, but does include any zero padding bytes at the end of a tile.</p> <p>Error Checking :</p> <p>1) only when tile bitstream has run out when the last block of the current tile has not been decoded. Set the error bit in MMIO, and perform error concealment to fill in the missing decoded block(s).</p> <p>2) there is no checking when there are bytes remaining after the last block of the current tile has been decoded.</p> <p>Note : HW AVP decoding pipeline is not required to parse the trailing bits and padding zeros at the end of a tile, tile group, and of a frame.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>[1,4294967295]</td> <td>Tile_Bitstream_Data_Length</td> <td>It has a valid range of 1 to 4294967295 (2³²-1) bytes. Zero byte length is not allowed. 2³¹ should be sufficient to handle a 16Kx16K, 12-bit, 4:4:4 frame size with a compression ratio of 1. Note : different bitstream LEVELs are having different requirement on the minimum compression ratio.</td> </tr> </tbody> </table>	Format:	U32	Value	Name	Description	[1,4294967295]	Tile_Bitstream_Data_Length	It has a valid range of 1 to 4294967295 (2 ³² -1) bytes. Zero byte length is not allowed. 2 ³¹ should be sufficient to handle a 16Kx16K, 12-bit, 4:4:4 frame size with a compression ratio of 1. Note : different bitstream LEVELs are having different requirement on the minimum compression ratio.
Format:	U32									
Value	Name	Description								
[1,4294967295]	Tile_Bitstream_Data_Length	It has a valid range of 1 to 4294967295 (2 ³² -1) bytes. Zero byte length is not allowed. 2 ³¹ should be sufficient to handle a 16Kx16K, 12-bit, 4:4:4 frame size with a compression ratio of 1. Note : different bitstream LEVELs are having different requirement on the minimum compression ratio.								
2	31:0	<p>Tile Indirect Data Start Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U32</td> </tr> </table> <p>Specifies the byte-aligned graphics memory starting address of a tile in the bitstream of a frame relative to the BSD Indirect Object Base Address. Each tile's coded bitstream is started and ended at a byte boundary. A frame with multiple tiles is coded as one bitstream. Indirect Data Start Address for each tile is equivalent to an address offset from the starting address of the frame bitstream.</p>	Format:	U32						
Format:	U32									

AVP_IND_OBJ_BASE_ADDR_STATE

AVP_IND_OBJ_BASE_ADDR_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The AVP Pipeline is selected with the Media Instruction Opcode "8h" for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The AVP_IND_OBJ_BASE_ADDR_STATE command is used to define the indirect object base address of the AV1 compressed bitstream in graphics memory. This is a frame level command issued in both encoding and decoding processes.</p> <p>Although a frame is coded as separate tiles, all tiles' compressed bitstream are still required to line up sequentially as one AV1 bitstream. Hence, there is only one Indirect Object Base Address for the entire AV1 coded frame. If the frame contains more than 1 tiles, the BSD Object Command will be issued multiple times, once for each tile and with its own tile bitstream starting memory address.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = AVP = 3h	
22:16	Media Instruction Command		
	Default Value:	3h AVP_IND_OBJ_BASE_ADDR_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
		Value	Name
	04h		
1..2	63:0	AVP Indirect Bitstream Object Base Address	
		Format:	SplitBaseAddress4KByteAligned

AVP_IND_OBJ_BASE_ADDR_STATE		
		Description
		Decoder: Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the AVP_BSD_OBJECT command for reading each tile's compressed bitstream in a frame.
3	31:0	AVP Indirect Bitstream Object Memory Address Attributes
		Format: MemoryAddressAttributes
4..5	63:0	AVP Indirect Bitstream Object Access Upper Bound
		Format: SplitBaseAddress4KByteAligned
		Decoder only.
		This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the AVP_BSD_OBJECT command for reading each tile's compressed bitstream in a frame. Indirect data accessed at this address or greater will cause the AVP to stop issuing requests to the GAC and the BSP VLD will then only receive zeros until a tile done is received.
		Setting this field to 0 will cause this range to be ignored by the AVP Pipeline.

AVP_INLOOP_FILTER_STATE

AVP_INLOOP_FILTER_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The AVP_INLOOP_FILTER_STATE command provides all the frame level syntax elements and derived parameters that are needed for the processing of all the post in-loop filters present in the AV1 codec, except the Luma and Chroma x0_qn which are tile based derived parameters. This includes the Deblocker, the CDEF (Constrained Directional Enhancement Filter), the HSRF (Horizontal-only Super-Resolution Filter), and the LRF (Loop Restoration Filter). These syntax elements can be changed in the bitstream from frame to frame.</p> <p>All Post In-Loop Filters are inherently frame-based filtering, but when implemented in a HW pipeline, the filtering process is performed in tile based and in a block by block fashion. In the addition to these frame and tile level states, there are additional syntax elements and derived parameters that are generated at SuperBlock level, and are not described here.</p> <p>Each of these 4 Post In-Loop Filters can be controlled independently and each can be enabled or disabled independently. Except the HSRF, all the other 3 filters have separate controls for each color plane as well. To disable a Post In-Loop Filter, its control parameter(s) are set to 0 - the default state.</p> <p>This command should be issued once per tile, even if no post in-loop filter is enabled for decoding the current frame. When in frame lossless mode or when IntraBC is enabled, all the Post In-Loop Filters are disabled for all color planes, this command will provide the default values for all parameters. All syntax elements are then assumed a value of 0, except otherwise specified in each field of this State Command.</p> <p>When it is in monochrome video, no filter parameter for the two chroma planes is present in the bitstream.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	Opcode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	Opcode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = AV1 = 3h	
	22:16	Media Instruction Command	
		Default Value:	33h AVP_INLOOP_FILTER_STATE
Format:		OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	

AVP_INLOOP_FILTER_STATE

		(Excludes Dwords 0, 1).	
		Value	Name
		Dh	
1	31	Deblocker Filter Delta LF Present Flag	
		Format:	U1
	<p>set to 1, specifies that additional loop filter delta values are present at the superblock level in the bitstream.</p> <p>set to 0, specifies that no additional loop filter delta values are present at the superblock level in the bitstream.</p> <p>It is the frame level syntax element delta_if_present flag. It is present in the bitstream, only if delta_q_present is set to 1. If delta_q_present is set to 0, delta_if_present flag is not present, and is defaulted to 0.</p>		
	30	Deblocker Filter Delta LF Multi Flag	
	Format:	U1	
<p>set to 1, specifies that at the Superblock level, separate loop filter deltas (multiple deltas) are sent for</p> <ol style="list-style-type: none"> 1) horizontal Luma Y edges, 2) vertical Luma edges, 3) both horizontal and vertical Chroma U edges, and 4) for both horizontal and vertical Chroma V edges. <p>set to 0, specifies that at the Superblock level, the same loop filter delta is used for all edges of all color planes (Y, U and V).</p> <p>It is the frame level syntax element delta_if_multi.</p> <p>It is present in the bitstream, when delta_if_present flag is set to 1. If delta_if_present flag is set to 0, delta_if_multi flag is not present, and is defaulted to 0.</p>			
29:28	Deblocker Delta LF Resolution		
	Format:	U2	
<p>It specifies the number of left shift which should be applied to decoded loop filter delta values. It is in the range of [0..3]. (no shift, and therefore no scaling).</p> <p>It is the frame level syntax element delta_if_res.</p> <p>It is present in the bitstream, when delta_if_present flag is set to 1. If delta_if_present flag is set to 0, delta_if_res is not present, and is defaulted to 0.</p> <p>Note : it is used to derive one part of the final filter levels: $lvl += read_delta_lflevel() * (1 \ll delta_if_res)$.</p> <p>Note : But in the reference C model, the same name is used for the derived parameter: $delta_if_res = 1 \ll (delta_if_res)$, which can take a 4-bit of value [1, 2, 4 or 8] instead.</p>			
27	Deblocker Filter Mode Ref Delta Enable Flag		
	Format:	U1	
<p>set to 1, means that the filter levels depends on the mode and reference frame used to predict a block.</p>			

AVP_INLOOP_FILTER_STATE			
	<p>set to 0, means that the filter level does not depend on the mode and reference frame. Default is 0.</p> <p>It is the frame level syntax element <code>loop_filter_delta_enabled</code>, or named <code>asmode_ref_delta_enabled</code>.</p>		
26:24	<p>Deblocker Filter Sharpness Level</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U3</td> </tr> </table> <p>It specifies the sharpness level of the deblocker. It is used to compute the deblocker filter limits (lim and mblim) for each of the 64 possible values of a filter level. The deblocker filter levels and the deblock filter sharpness together determine when a block edge is filtered, and by how much the filtering can change the sample values.</p> <p>It is the frame level syntax element <code>loop_filter_sharpness</code>, or named as <code>sharpness_level</code>. It is in the range of [0..7]. Default is 0.</p>	Format:	U3
Format:	U3		
23:18	<p>Chroma V Deblocker Filter Level</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>It specifies the deblocker filter strength for both the horizontal (using vertical filters) and the vertical (using horizontal filters) edges of the Chroma Vplane.</p> <p>It is the frame level syntax element <code>loop_filter_level[3]</code>, or named as <code>filter_level_v</code>. It is present, [only if <code>filter_level[0]</code> and <code>filter_level[1]</code> are not both set to 0 AND only if the current frame is not a monochrome]. If not present, it is default to 0.</p> <p>It is in the range of [0..63]. Default is 0.</p>	Format:	U6
Format:	U6		
17:12	<p>Chroma U Deblocker Filter Level</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>It specifies the deblocker filter strength for both the horizontal (using vertical filters) and the vertical (using horizontal filters) edges of the Chroma U plane.</p> <p>It is the frame level syntax element <code>loop_filter_level[2]</code>, or named as <code>filter_level_u</code>. It is present, [only if <code>filter_level[0]</code> and <code>filter_level[1]</code> are not both set to 0 AND only if the current frame is not a monochrome]. If not present, it is default to 0.</p> <p>It is in the range of [0..63]. Default is 0.</p>	Format:	U6
Format:	U6		
11:6	<p>Luma Y Deblocker Filter Level Horizontal</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>It specifies the deblocker filter strength for the horizontal edges (using vertical filters) of the Luma plane.</p> <p>It is the frame syntax element <code>loop_filter_level[1]</code>.</p> <p>It is in the range of [0..63]. Default is 0.</p> <p>Setting <code>loop_filter_level[0] = loop_filter_level[1] = 0</code>, disables the deblocker filter.</p>	Format:	U6
Format:	U6		
5:0	<p>Luma Y Deblocker Filter Level Vertical</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>It specifies the deblocker filter strength for the vertical edges (using horizontal filters) of the</p>	Format:	U6
Format:	U6		

AVP_INLOOP_FILTER_STATE				
		<p>Luma plane. It is the frame syntax element <code>loop_filter_level[0]</code>. It is in the range of [0..63]. Default is 0. Setting <code>loop_filter_level[0] = loop_filter_level[1] = 0</code>, disables the deblocker filter.</p>		
2	31	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	30:24	<p>Deblocker Filter Ref Deltas[3]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 by <code>setup_past_independence()</code>. If this syntax element is not present in the bitstream (i.e. <code>update_ref_delta=0</code>), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].</p>	Format:	S6
	Format:	S6		
	23	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	22:16	<p>Deblocker Filter Ref Deltas[2]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 by <code>setup_past_independence()</code>. If this syntax element is not present in the bitstream (i.e. <code>update_ref_delta=0</code>), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].</p>	Format:	S6
	Format:	S6		
15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
14:8	<p>Deblocker Filter Ref Deltas[1]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 by <code>setup_past_independence()</code>. If this syntax element is not present in the bitstream (i.e. <code>update_ref_delta=0</code>), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].</p>	Format:	S6	
Format:	S6			
7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
6:0	<p>Deblocker Filter Ref Deltas[0]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It specifies the adjustment needed for the deblocker filter level based on which one of the 8 possible reference frames is in used - Deblocker Filter Ref Deltas[Ref=0 to 7]. Ref=0 to 7 is defined as follows:</p>	Format:	S6	
Format:	S6			

AVP_INLOOP_FILTER_STATE

			<p>[0] = INTRA_FRAME [1] = LAST_FRAME [2] = LAST2_FRAME [3] = LAST3_FRAME [4] = GOLDEN_FRAME [5] = BWDREF_FRAME [6] = ALTREF2_FRAME [7] = ALTREF_FRAME</p> <p>Deblocker Filter Ref Deltas[] is used to pre-compute the final deblocker filter level for all blocks within a segment, $lv[seg_id][horz_or_vert][ref_frame[0]][block_coding_mode]$. The pre-compute can take place either at the frame header (if $\Delta_{lf_present_flag} = 0$), or at the block level (if $\Delta_{lf_present_flag} = 0$); but the equations and clamping used are different in these 2 cases. It is the frame level syntax element $loop_filter_ref_deltas[Ref=0\ to\ 7]$, or named as $ref_deltas[Ref=0\ to\ 7]$.</p> <p>It is in 7-bits 2's complement within the range of [-64 to +63]. Initialize to $1 \times setup_past_independence()$.</p> <p>If this syntax element is not present in the bitstream (i.e. $update_ref_delta=0$), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame.</p> <p>Note : The 8 elements of the Deblock Filter Ref Deltas[Ref=0 to 7] are initialized differently inside the $setup_past_independence()$.</p> <p>Deblock Filter Ref Deltas[INTRA_FRAME] is set equal to 1. Deblock Filter Ref Deltas[LAST_FRAME] is set equal to 0. Deblock Filter Ref Deltas[LAST2_FRAME] is set equal to 0. Deblock Filter Ref Deltas[LAST3_FRAME] is set equal to 0. Deblock Filter Ref Deltas[BWDREF_FRAME] is set equal to 0. Deblock Filter Ref Deltas[GOLDEN_FRAME] is set equal to -1. Deblock Filter Ref Deltas[ALTREF_FRAME] is set equal to -1. Deblock Filter Ref Deltas[ALTREF2_FRAME] is set equal to -1.</p>		
3	31	Reserved	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">Format:</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ			
	30:24	Deblocker Filter Ref Deltas[7]	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">Format:</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64 to +63]. Initialize to $-1 \times setup_past_independence()$.</p> <p>If this syntax element is not present in the bitstream (i.e. $update_ref_delta=0$), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame.</p> <p>Refer to the full description in Deblocker Filter Ref Deltas[0].</p>	Format:	S6
	Format:	S6			
23	Reserved	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">Format:</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
22:16	Deblocker Filter Ref Deltas[6]	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">Format:</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64 to +63]. Initialize to -</p>	Format:	S6	
Format:	S6				

AVP_INLOOP_FILTER_STATE				
		<p>1bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].</p>		
	15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
	14:8	<p>Deblocker Filter Ref Deltas[5]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to -1bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].</p>	Format:	S6
Format:	S6			
	7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
	6:0	<p>Deblocker Filter Ref Deltas[4]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].</p>	Format:	S6
Format:	S6			
4	31:15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	14:8	<p>Deblocker Filter Mode Deltas[1]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_mode_delta=0), Deblocker Filter ModeDeltas[BPM=0 to 1] maintains its value from previous frame. Refer to the full description in Deblocker Filter Mode Deltas[0].</p>	Format:	S6
	Format:	S6		
7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
6:0	<p>Deblocker Filter Mode Deltas[0]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>S6</td> </tr> </table> <p>It specifies the adjustment needed for the deblocker filter level based on which block prediction mode is in used - Deblocker Filter Mode Deltas[BPM=0 to 1]. BPM=0 to 1 is defined by mode_if_lut[Block Prediction Mode] ; Block Prediction Mode can be one</p>	Format:	S6	
Format:	S6			

AVP_INLOOP_FILTER_STATE

		<p>of the 25 possible block prediction modes being defined in AV1.</p> <p>BPM=0, for all 13 intra block prediction modes, for the GLOBALMV inter block prediction mode, and for the GLOBAL_GLOBALMV inter compound block prediction mode.</p> <p>BPM =1, for all the other 10 inter block prediction modes (3 inter block prediction modes and 7 inter compound block prediction modes)</p> <p>Deblocker Filter Mode Deltas[] is used to pre-compute the final deblocker filter level for all blocks within a segment,</p> <p>$lv[seg_id][horz_or_vert][ref_frame[0]][mode_lf_lut[block_prediction_mode]]$. The pre-compute can take place either at the frame header (if $\Delta lf_present_flag = 0$), or at the block level (if $\Delta lf_present_flag \neq 0$); but the equations and clamping used are different in these 2 cases.</p> <p>It is the frame level syntax element $loop_filter_mode_deltas[BPM=0\ to\ 1]$, or named as $mode_deltas[BPM=0\ to\ 1]$.</p> <p>It is in 7-bits 2's complement within the range of [-64 to +63]. Initialize to 0 by <code>setup_past_independence()</code>.</p> <p>If this syntax element is not present in the bitstream (i.e. $update_mode_delta=0$), Deblocker Filter Mode Deltas[BPM=0 to 1] maintains its value from previous frame.</p>		
5	31:30	<p>CDEF Filter Damping Factor Minus3</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It specifies the amount of damping in the deringing filter.</p> <p>It is the frame level syntax element, $cdef_damping_minus_3$.</p> <p>It is also named as $cdef_damping (=cdef_damping_minus3+3)$.</p> <p>It takes on value in the range of [0 to 3]. Default is 0.</p>	Format:	U2
	Format:	U2		
	29:28	<p>CDEF Bits</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It is used for 2 purposes:</p> <ol style="list-style-type: none"> 1) $[1 \ll cdef_bits]$ specifies the number of frame level CDEF Y and UV strengths to be read from the bitstream in the uncompressed header. 2) $cdef_bits$ specifies the number of bits in the bitstream, at the block level, that need to be read for the CDEF strength to be applied. <p>It is the frame level syntax element, $cdef_bits$.</p> <p>It is in the range of [0..3]. Default is 0.</p> <p>Note: to disable CDEF Filtering process, set $cdef_bits$ to 0, and set $cdef_y_strengths[0]=cdef_uv_strengths[0]=0$.</p>	Format:	U2
	Format:	U2		
	27:24	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
23:18	<p>CDEF Y Strength[3]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>Refer to the full description in CDEF Y Strength[0].</p>	Format:	U6	
Format:	U6			
17:12	<p>CDEF Y Strength[2]</p>			

AVP_INLOOP_FILTER_STATE

		Format:	U6
		Refer to the full description in CDEF Y Strength[0].	
	11:6	CDEF Y Strength[1]	
		Format:	U6
		Refer to the full description in CDEF Y Strength[0].	
	5:0	CDEF Y Strength[0]	
		Format:	U6
		<p>It specifies one of the 8 possible filter strengths for the CDEF Luma Y Filter.</p> <p>It is the frame level syntax element <code>cdef_y_strength[i=0 to 7]</code>, or also named simply as <code>cdef_strengths[i=0 to 7]</code>.</p> <p>The number of active filter strengths in used for the current frame is equal to $[1 \ll cdef_bits]$, which can only be [1, 2, 4 or 8].</p> <p>Each strength is in the range of [0 to 63]. For all filter strengths that are not present in the bitstream, they are defaulted to 0.</p> <p>Note: to disable CDEF Filtering process, set <code>cdef_bits</code> to 0, and set <code>cdef_y_strengths[0]=cdef_uv_strengths[0]=0</code>.</p>	
6	31:24	Reserved	
		Format:	MBZ
	23:18	CDEF Y Strength[7]	
		Format:	U6
		Refer to the full description in CDEF Y Strength[0].	
	17:12	CDEF Y Strength[6]	
		Format:	U6
		Refer to the full description in CDEF Y Strength[0].	
	11:6	CDEF Y Strength[5]	
		Format:	U6
		Refer to the full description in CDEF Y Strength[0].	
	5:0	CDEF Y Strength[4]	
		Format:	U6
		Refer to the full description in CDEF Y Strength[0].	
7	31:24	Reserved	
		Format:	MBZ
	23:18	CDEF UV Strength[3]	
		Format:	U6

AVP_INLOOP_FILTER_STATE				
		Refer to the full description in CDEF UVStrength[0].		
	17:12	CDEF UV Strength[2] <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> Refer to the full description in CDEF UVStrength[0].	Format:	U6
Format:	U6			
	11:6	CDEF UV Strength[1] <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> Refer to the full description in CDEF UVStrength[0].	Format:	U6
Format:	U6			
	5:0	CDEF UV Strength[0] <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> <p>It specifies one of the 8 possible filter strengths for the CDEF Chroma U/VFilter. Chroma U and V share the same strength specification.</p> <p>It is the frame level syntax element <code>cdef_uv_strength[i=0 to 7]</code>, or named as <code>cdef_uv_strengths[i=0 to 7]</code>.</p> <p>The number of active filter strengths in used for the current frame is equal to $[1 \ll cdef_bits]$, which can only be [1, 2, 4 or 8]. For monochrome video, there is no Chroma UV strength in the bitstream, either.</p> <p>Each strength is in the range of [0to 63]. The 7th and 8th-bit are ignored. For all filter strengths that are not present in the bitstream, they are defaulted to 0.</p> <p>Note: to disable CDEF Filtering process, set <code>cdef_bits</code> to 0, and set <code>cdef_y_strengths[0]=cdef_uv_strengths[0]=0</code>.</p>	Format:	U6
Format:	U6			
8	31:24	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	23:18	CDEF UV Strength[7] <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> Refer to the full description in CDEF UVStrength[0].	Format:	U6
	Format:	U6		
	17:12	CDEF UV Strength[6] <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> Refer to the full description in CDEF UVStrength[0].	Format:	U6
Format:	U6			
11:6	CDEF UV Strength[5] <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> Refer to the full description in CDEF UVStrength[0].	Format:	U6	
Format:	U6			
5:0	CDEF UV Strength[4] <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> Refer to the full description in CDEF UVStrength[0].	Format:	U6	
Format:	U6			

AVP_INLOOP_FILTER_STATE

9	31:21	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
20:16	Super-Res Denom <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U5</td> </tr> </table> <p>It specifies an integer denominator of a fixed point fractional number (scaling factor) which is used to scale down the current frame width (in pixels) for the subsequent block level decoding process of its bitstream. Super-resolution is only applied to the horizontal direction, so this scaling factor is applied to the frame width only.</p> <p>The numerator of this down-scaling factor is always set to 8.</p> <p>The reduced (scaled down) frame width in pixels = (upscaled frame width in pixels * 8 + (super-res denom»1)) / super-res denom. This division is done by Driver.</p> <p>It is derived from the 3-bits frame level syntax element, coded_denom. Super-res denom = coded_denom + 9, which is always > 8 (i.e. downscaling factor is always < 1.0).</p> <p>That is, the super-resolution down-scaling factor can go from [8/9 to 8/16], and the corresponding up-scaling factor can go from [9/8 to 16/8].</p> <p>If super-resolution is not enabled, super-res denom is not present in the bitstream, its value is defaulted to 8 (i.e. no scaling). Hence, Super-Res Denom is in the range of [8.. 16]. Value 0 to 7 are not allowed.</p> <p>Different frames in a video sequence can have different super-res denom. Adjacent frames can all have different super-res denoms.</p> <p>Super-res denom is used in multiple functions of the Super-resolution and the Loop Restoration processes, but no fixed-point division is needed.</p> <p>Note : if not monochrome video, the same scaling factor and super-resolution process are applied to the two chroma planes as well; otherwise, only the Luma plane is being processed.</p> <p>Following restrictions apply in Encoder Mode if Wiener Filter enabled:</p> <p>In Single pipe: Only valid values are 10, 12, 14 and 16</p> <p>In Multi pipe: Only valid value is 16</p> <p>otherwise all values are valid..</p>	Format:	U5	
Format:	U5			
	15:0	Super-Res Upscaled Frame Width Minus1 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>It specifies the super-resolution upscaled frame width in pixels, without padding on the right and bottom of the frame. Since, super-resolution is only applied to the horizontal frame width, there is no scaling to the vertical frame height.</p> <p>The input and output of the Loop Restoration Filter and all reference frames in the Display Buffer (DPB) are in Super-Res Upscaled Frame Width.</p> <p>When super-resolution is enabled, the bitstream is coded with the reduced frame width. Hence, the current frame width programmed in the AVP_PIC_STATE is the derived reduced frame width without padding on the right and bottom border of the frame.</p> <p>The scaled down frame width (AVP_PIC_STATE frame width minus1 + 1)= (upscaled frame width * 8 + (super-res denom»1)) / super-res denom.</p> <p>Upscaled frame width is derived as follows:</p> <p>A) In KEY FRAME or INTRA-ONLY NON-KEY FRAME:</p> <p>1) when frame_size_override_flag is set to 1, it is the frame level syntax element frame_width_minus1.</p>	Format:	U16
Format:	U16			

AVP_INLOOP_FILTER_STATE				
	<p>2) when frame_size_override_flag is set to 1, it is the sequence level syntax element max_frame_width_minus1.</p> <p>B) In INTER FRAME:</p> <p>1) if the current frame size can be inferred from one of the 7 possible reference frames, it is the derived frame width minus1. Otherwise,</p> <p>2) when frame_size_override_flag is set to 1, it is the frame level syntax element frame_width_minus1.</p> <p>3) when frame_size_override_flag is set to 1, it is the sequence level syntax element max_frame_width_minus1.</p> <p>Max frame size is 64K, but intel only support up to 16K. Hence, bit 14 and 15 are not used.</p> <p>Default is 7, for minimum frame width is 8 pixels.</p> <p>Note : if upscaling factor > 1.0, AVP_PIC_STATE Frame Width must < 16K.</p> <p>Note : both the upscaled frame width and the downscaled frame width are provided to the AVP HW pipeline, so that HW does not need to perform the division in the scaling process.</p>			
10	31:11	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	10	<p>Use Same Loop Restoration Unit Size for Chromas UV Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>It specifies how the Chroma UV Loop Restoration Unit size should be derived from that of the Luma. Chroma U and Chroma V are having the same LRU Size.</p> <p>Set to 1, if Chroma LRU size is the same as that of the Luma.</p> <p>Set to 0, if Chroma LRU size is subsampled that of the Luma in 4:2:0 by half in each dimension.</p> <p>Default is 0.</p> <p>It is the frame level syntax element, lr_uv_shift, which is present only if current video is a 4:2:0 and not both Chroma plane having filter type set to RESTORE_NONE.</p> <p>For 4:2:2 and 4:4:4, LRU size for Chromas UV is always the same as that of Luma.</p> <p>For monochrome, this field is ignored.</p> <p>Intel Encoder PAK only support LRU size = SuperBlock size = 64x64 pixels, this field must also be set to 0.</p>	Format:	U1
Format:	U1			
9:8	<p>Loop Restoration Unit Size for Luma Y</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It specifies the Loop Restoration Unit size in pixels for Luma Y.</p> <p>Loop Restoration Unit is a square grid in the current frame for each color plane. Each Loop Restoration Unit of a color plane has its own set of Loop Restoration Filter parameters.</p> <p>It is derived from at most 2 1-bit syntax element (lr_unit_shift) in the frame header which are present only if not all 3 color planes having filter type set to RESTORE_NONE. If all 3 color planes having filter type set to RESTORE_NONE, then Loop Restoration Unit Size for Luma plane is defaulted to 256x256.</p> <p>It is also named as LoopRestorationSize[i=0], or restoration_unit_size for Luma plane. But intel has mapped them to the following code:</p> <p>0, for 0 size (Default, when LR is not enabled)</p> <p>1, for 64x64 pixels LRU nominal size.</p> <p>2, for 128x128 pixels LRU nominal size.</p>	Format:	U2	
Format:	U2			

AVP_INLOOP_FILTER_STATE

	<p>3, for 256x256 pixels LRU nominal size. Intel Encoder PAK only support LRU size = SuperBlock size = 64x64 pixels. Note : LRU Luma Size cannot < Superblock Size. If Superblock Size is 128x128, LRU Luma Size cannot be 64x64.</p>		
7:6	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
5:4	<p>Frame Loop Restoration Filter Type for Chroma V</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">U2</td> </tr> </table> <p>It specifies the Frame Level Loop Restoration Filter Type for the Chroma V plane. It is derived from 2 bits syntax element (lr_type) in the frame header, if not in a monochrome video. It is also named as framerestorationtype[i=2], or frame_restoration_type for Chroma V plane. It is in the range of [0..3]. For a monochrome video, it is defaulted to 0. 0, for RESTORE_NONE. 1, for RESTORE_WIENER. 2, for RESTORE_SGRPROJ (Dual Self-Guided Projection Filter). 3, for RESTORE_SWITCHABLE (LRU level choice of NONE, WIENER or SGRPROJ). Each color plane can have its own Frame Level Loop Restoration Filter Type specification. Note : to disable Loop Restoration Filtering in the current frame, the Frame Restoration Filter Type for all color planes must be set to RESTORE_NONE. Note : that the syntax element lr_type uses a different enum order for the four LR filter types (0 for RESTORE_NONE, 1 for SWITCHABLE, 2 for WIENER, and 3 for SGRPROJ).</p>	Format:	U2
Format:	U2		
3:2	<p>Frame Loop Restoration Filter Type for Chroma U</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">U2</td> </tr> </table> <p>It specifies the Frame Level Loop Restoration Filter Type for the Chroma U plane. It is derived from 2 bits syntax element (lr_type) in the frame header, if not in a monochrome video. It is also named as framerestorationtype[i=1], or frame_restoration_type for Chroma U plane. It is in the range of [0..3]. For a monochrome video, it is defaulted to 0. 0, for RESTORE_NONE. 1, for RESTORE_WIENER. 2, for RESTORE_SGRPROJ (Dual Self-Guided Projection Filter). 3, for RESTORE_SWITCHABLE (LRU level choice of NONE, WIENER or SGRPROJ). Each color plane can have its own Frame Level Loop Restoration Filter Type specification. Note : to disable Loop Restoration Filtering in the current frame, the Frame Restoration Filter Type for all color planes must be set to RESTORE_NONE. Note : that the syntax element lr_type uses a different enum order for the four LR filter types (0 for RESTORE_NONE, 1 for SWITCHABLE, 2 for WIENER, and 3 for SGRPROJ).</p>	Format:	U2
Format:	U2		
1:0	<p>Frame Loop Restoration Filter Type for Luma Y</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">U2</td> </tr> </table> <p>It specifies the Frame Level Loop Restoration Filter Type for the Luma plane.</p>	Format:	U2
Format:	U2		

AVP_INLOOP_FILTER_STATE				
		<p>It is derived from 2 bits syntax element (lr_type) in the frame header. It is also named as framerestorationtype[i=0], or frame_restoration_type for Luma plane.</p> <p>It is in the range of [0..3]. Default is 0.</p> <p>0, for RESTORE_NONE.</p> <p>1, for RESTORE_WIENER.</p> <p>2, for RESTORE_SGRPROJ (Dual Self-Guided Projection Filter).</p> <p>3, for RESTORE_SWITCHABLE (LRU level choice of NONE, WIENER or SGRPROJ).</p> <p>Each color plane can have its own Frame Level Loop Restoration Filter Type specification.</p> <p>Note : to disable Loop Restoration Filtering in the current frame, the Frame Restoration Filter Type for all color planes must be set to RESTORE_NONE.</p> <p>Note : that the syntax element lr_type uses a different enum order for the four LR filter types (0 for RESTORE_NONE, 1 for SWITCHABLE, 2 for WIENER, and 3 for SGRPROJ).</p>		
11	31:16	<p>Reserved (for higher precision of x_step_qn)</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	<p>Luma Plane x_step_qn</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U16</td> </tr> </table> <p>Derived parameter, specifies the increment of the super-res luma upscaling step in pixel position for the current frame.</p>	Format:	U16	
Format:	U16			
12	31:0	<p>Luma Plane x0_qn</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>S31</td> </tr> </table> <p>Derived parameters, specifies the starting offset (in 2's complement signed integer) of the super-res upscaling process for each tile in the original frame for Luma.</p>	Format:	S31
Format:	S31			
13	31:16	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	<p>Chroma Plane x_step_qn</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U16</td> </tr> </table> <p>Derived parameter, specifies the increment of the super-res chroma upscaling step in pixel position for the current frame.</p>	Format:	U16	
Format:	U16			
14	31:0	<p>Chroma Plane x0_qn</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>S31</td> </tr> </table> <p>Derived parameters, specifies the starting offset (in 2's complement signed integer) of the super-res upscaling process for each tile in the original frame for chroma.</p>	Format:	S31
Format:	S31			

AVP_INTER_PRED_STATE

AVP_INTER_PRED_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The AVP Pipeline is selected with the Media Instruction Opcode "8h" for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16. AVP supports a 8-reference frames display buffer. But at any given frame being decoded, only up to 7reference frames out of the 8 can be active. There are also further constraints on which of these 7frames can be used for forward and backward reference in the compound mode.</p> <p>To simplify the decoder command sequence programming, this command is issued once for each inter-coded tile as well as for each intra-coded tile (such as in a KEY_FRAME, a DELAY_KEY_FRAME, an INTRA_ONLY_FRAME).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = AVP = 3h	
22:16	Media Instruction Command		
	Default Value:	12h AVP_INTER_PRED_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
1	31:24	Saved Order Hints for All References[0][3]	
		Format:	U8
	23:16	Saved Order Hints for All References[0][2]	
		Format:	U8

AVP_INTER_PRED_STATE		
	15:8	Saved Order Hints for All References[0][1] Format: U8
	7:0	Saved Order Hints for All References[0][0] Format: U8 <p>This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.</p>
2	31:24	Active Reference Bitmask for Motion Field Projection Format: U8 <p>This field specifies which ones of the 7 references are involved in the Motion Field Projection. bit 0 corresponding to the LAST reference frame for the decoding current frame. bit 6 corresponding to the ALTREF reference frame for decoding the current frame. bit 7 is reserved and must set to 0. This is an intel derived parameters. A reference has its corresponding bit in the bitmask set to 1 if 1) motion_field_projection() is called for that reference from av1_setip_motion_field(); AND 2) motion_field_projection() for that reference does not return 0, meaning the following conditions have to be satisfied for that reference: a) Reference is available (i.e. frame buffer index has a valid value) b) Reference is not intra-only (i.e. neither a KEY FRAME nor an INTRA-ONLY FRAME) c) Reference is not scaled. Otherwise set to 0.</p>
	23:16	Saved Order Hints for All References[0][6] Format: U8
	15:8	Saved Order Hints for All References[0][5] Format: U8
	7:0	Saved Order Hints for All References[0][4] Format: U8
	3	31:24

AVP_INTER_PRED_STATE			
	23:16	Saved Order Hints for All References[1][2]	Format: U8
	15:8	Saved Order Hints for All References[1][1]	Format: U8
	7:0	Saved Order Hints for All References[1][0]	Format: U8
		<p>This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections.</p> <p>Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame.</p> <p>Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here.</p> <p>Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value).</p> <p>Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.</p>	
4	31:24	Reserved	Format: MBZ
	23:16	Saved Order Hints for All References[1][6]	Format: U8
	15:8	Saved Order Hints for All References[1][5]	Format: U8
	7:0	Saved Order Hints for All References[1][4]	Format: U8
5	31:24	Saved Order Hints for All References[2][3]	Format: U8
	23:16	Saved Order Hints for All References[2][2]	Format: U8
	15:8	Saved Order Hints for All References[2][1]	Format: U8
	7:0	Saved Order Hints for All References[2][0]	Format: U8
		<p>This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections.</p> <p>Array indices are defined as [a reference frame index][the 7 reference indices of the reference</p>	

AVP_INTER_PRED_STATE			
	<p>frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame.</p> <p>Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here.</p> <p>Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value).</p> <p>Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.</p>		
6	31:24	Reserved	Format: MBZ
	23:16	Saved Order Hints for All References[2][6]	Format: U8
	15:8	Saved Order Hints for All References[2][5]	Format: U8
	7:0	Saved Order Hints for All References[2][4]	Format: U8
7	31:24	Saved Order Hints for All References[3][3]	Format: U8
	23:16	Saved Order Hints for All References[3][2]	Format: U8
	15:8	Saved Order Hints for All References[3][1]	Format: U8
	7:0	Saved Order Hints for All References[3][0]	Format: U8
	<p>This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections.</p> <p>Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame.</p> <p>Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here.</p> <p>Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value).</p> <p>Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.</p>		

AVP_INTER_PRED_STATE		
8	31:24	Reserved Format: MBZ
	23:16	Saved Order Hints for All References[3][6] Format: U8
	15:8	Saved Order Hints for All References[3][5] Format: U8
	7:0	Saved Order Hints for All References[3][4] Format: U8
9	31:24	Saved Order Hints for All References[4][3] Format: U8
	23:16	Saved Order Hints for All References[4][2] Format: U8
	15:8	Saved Order Hints for All References[4][1] Format: U8
	7:0	Saved Order Hints for All References[4][0] Format: U8 <p>This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (5-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.</p>
10	31:24	Reserved Format: MBZ
	23:16	Saved Order Hints for All References[4][6] Format: U8
	15:8	Saved Order Hints for All References[4][5] Format: U8
	7:0	Saved Order Hints for All References[4][4]

AVP_INTER_PRED_STATE		
		Format: U8
11	31:24	Saved Order Hints for All References[5][3] Format: U8
	23:16	Saved Order Hints for All References[5][2] Format: U8
	15:8	Saved Order Hints for All References[5][1] Format: U8
	7:0	Saved Order Hints for All References[5][0] Format: U8
	<p>This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections.</p> <p>Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame.</p> <p>Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here.</p> <p>Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value).</p> <p>Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.</p>	
12	31:24	Reserved Format: MBZ
	23:16	Saved Order Hints for All References[5][6] Format: U8
	15:8	Saved Order Hints for All References[5][5] Format: U8
	7:0	Saved Order Hints for All References[5][4] Format: U8
13	31:24	Saved Order Hints for All References[6][3] Format: U8
	23:16	Saved Order Hints for All References[6][2] Format: U8
	15:8	Saved Order Hints for All References[6][1] Format: U8
	7:0	Saved Order Hints for All References[6][0]

AVP_INTER_PRED_STATE				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections.</p> <p>Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame.</p> <p>Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here.</p> <p>Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value).</p> <p>Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.</p>	Format:	U8
Format:	U8			
14	31:24	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	23:16	Saved Order Hints for All References[6][6] <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table>	Format:	U8
	Format:	U8		
15:8	Saved Order Hints for All References[6][5] <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table>	Format:	U8	
Format:	U8			
7:0	Saved Order Hints for All References[6][4] <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table>	Format:	U8	
Format:	U8			

AVP_PIC_STATE

AVP_PIC_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>All AVP_PIC_STATE should stay the same for the whole frame even if AVP_PIC_STATE is re-programmed for every tiles.</p> <p>The bitfields of AVP_PIC_STATE are defined either from</p> <p>1) syntax elements of the uncompressed sequence header (received from sequence_header_obu) and of the uncompressed frame header (received from frame_header_obu),</p> <p>2) or, parameters derived from 1).</p> <p>Note : Bitstreams may contain several copies of the frame header (there can only be one frame_header_obu, but multiple redundant_frame_header_obu)interspersed with tile_group_obu to allow for greater error resilience. However, the copies must contain identical contents to the original frame_header_obu.</p> <p>Note : there should be only one sequence_header_obu per video sequence.</p> <p>Note : AVP pipeline is invoked to decode a frame from the bitstream, only if that frame has show_existing_frame flag (syntax element in the frame header) set to 0. For the case thatshow_existing_frame flag is set to 1, application and driver process the frame instead, no block level decoding is needed.</p> <p>Note : Unlike VP9, AV1 does not have a compressed header. All the syntax elements defined in the AV1 sequence and frame level headers are not arithmetic coded, hence application and driver can directly read them off from the bitstream. If encryption is ON, then only application can parse the sequence and frame headers.</p> <p>Note : the values of the sequence header/level syntax elements and their derived parameters are to last throughout all frames in the video sequence, until the next Sequence Header OBU is received that may change them. But some sequence header/level syntax elements or their derived parameters may further qualified by frame header/level syntax elements and their derived parameters, then these type of syntax elements and their derived parameters can be changed frame to frame.</p> <p>Note : the values of the frame header/level syntax elements and their derived parameters can be changed from frame to frame.</p> <p>Note : there are some syntax elements and their derived parameters can be changed only at KEY FRAME. Hence, the values of these type of syntax elements and their derived parameters can last for the entire GOP, i.e. until the next KEY FRAME that may change them.</p> <p>Note : there is no separate profile for Still Picture. Still Picture is coded and decoded as a KEY FRAME, with all coding tools supported (tiling, all post in-loop filters, film grain injection, monochrome, intraBC, palette prediction mode, etc.). There is no restriction in coding Still Picture as a KEY FRAME.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
28:27	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
26:23	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name

AVP_PIC_STATE				
		<table border="1"> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table> <p>Codec/Engine Name = AVP = 3h</p>	Format:	OpCode
Format:	OpCode			
22:16	Media Instruction Command			
	Default Value:	30h AVP_PIC_STATE		
	Format:	OpCode		
15:12	Reserved			
	Format:	MBZ		
11:0	Dword Length			
	Format:	=n		
	(Excludes Dwords 0, 1).			
	Value	Name Programming Notes		
	31h	Decoder DW Length Only Up to DW12 should be programmed for decoder		
	48h	Encoder DW Length All DWs should be programmed for encoder		
1	31:16	Frame Height In Pixel Minus 1		
		<table border="1"> <tr> <td>Format:</td> <td>U16</td> </tr> </table> <p>Specifies the height of the frame to be decoded in unit of pixel. It is the same as the frame height in luma samples.</p> <p>AV1 supports up to 64Kx64K frame size. Intel supports up to 16Kx16K frame size. Valid range [15, 16383].</p> <p>Min frame height is 16pixels. Hence, Luma is 16and Chroma is 8(in 4:2:0).</p> <p>It is a frame-level derived parameters, after taking into account of the syntax elementmax_frame_height in the sequence header and frame_size_override_flag in the frame header. For SWITCH Frame,frame_size_override_flag is always set to 1; for all other frame types (KEY Frame, INTRA-ONLY Frame and INTER Frame),frame_size_override_flag is read from the bitstream (and can be 0 or 1).For inter-frame, it also takes into account of the possibility of using frame size inferred from a reference frame.</p> <p>Note : this field is not affected by Horizontal Super-Resolution coding.</p> <p>Note : this frame height may be odd or even number, and may not be divisible by 4, 8 , superblock size, tile size, or LRU size.</p> <p>Note : if frame height is an odd number, the corresponding height of the Chroma planes in 4:2:0 is rounded up. For example, if Luma height is 13 pixels, Chroma height is 7 pixels.</p> <p>Note : internally the frame height is rounded up to be divisible by 8 in both the HW encoder and decoder. The padding in the encoder side is not normative, but will be coded into the bitstream. When decoder reconstructs the decoded frame from the bitstream, the padding is being reconstructed as well. The padding at the right and bottom borders of the decoded frame can be cropped away before display. But they are not cropped away when the decoded frame is added into the DPB as a reference frame.</p> <p>Note : this is NOT the Render Frame Height, which is used to crop the decoded frame size for display.</p>	Format:	U16
Format:	U16			
	15:0	Frame Width In Pixel Minus 1		

AVP_PIC_STATE

		Format:	U16
		<p>Specifies the width of the decoded frame in unit of pixel. It is the same as the frame width in uma samples.</p> <p>AV1 supports up to 64Kx64K frame size. Intel supports up to 16Kx16K frame size.</p> <p>Valid range [15, 16383].</p> <p>Min frame width is 16pixels. Hence, Luma is 16and Chroma is 8(in 4:2:0).</p> <p>It is a frame-level derived parameters, after taking into account of the syntax elementmax_frame_width in the sequence header and frame_size_override_flag in the frame header. For inter-frame and s-frame, it also takes into account of the possibility of using frame size inferred from a reference frame.</p> <p>When the Horizontal Super-Resolution is active, this frame width is the downscaled frame width. Note : if Horizontal-Only Super-Resolution is ON, this field specifies the reduced width of the downscaled frame size. The number of Superblocks to be processed per row in this case is derived from this reduced frame width. The tile configuration is also specified in this reduced frame width. But after deblocker and CDEFfiltering (if either or both are active), this reduced frame width is scaled back up to the original frame width for the subsequent Loop Restoration filtering (if active), which then is outputted for display or become a reference frame. Hence, all reference frames in the DPB are always stored in full frame resolution, and dynamic on-the-fly frame resizing is always invoked during Motion Comp at block level, which is coded with the reduced frame width.</p> <p>Note that the upscaled frame width cannot exist 16K range. upscaling frame width = reduced frame width * upscaling factor. The upscaled frame width is provided in the AVP_INLOOP_FILTER_STATE Command.</p> <p>Note that this frame width may be odd or even number, and may not be divisible by 4, 8 , superblock size, tile size, or LRU size.</p> <p>Note : if frame width is an odd number, the corresponding width of the Chroma planes in 4:2:0 is rounded up. For example, if Luma width is 13 pixels, Chroma width is 7 pixels.</p> <p>Note : internally the frame width is rounded up to be divisible by 8 in both the HW encoder and decoder. The padding in the encoder side is not normative, but will be coded into the bitstream. When decoder reconstructs the decoded frame from the bitstream, the padding is being reconstructed as well. The padding at the right and bottom borders of the decoded frame can be cropped away before display. But they are not cropped away when the decoded frame is added into the DPB as a reference frame.</p> <p>Note : this is NOT the Render Frame Width, which is used to crop the decoded frame size for display.</p>	
2	31:25	Reserved	
		Format:	MBZ
	24:22	Reserved	
		Format:	MBZ
	21	Sequence Enable Joint Compound Flag	
		Format:	U1
		<p>It specifies whether the Joint Compound coding tool is enabled for the video sequence, or not. It is the sequence level syntax element, enable_jnt_comp, which is only present in the bitstream</p>	

AVP_PIC_STATE											
		<p>when the sequence level syntax element enable_order_hint is set to 1. It is defaulted to 0, when enable_order_hint is set to 0.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>[Default]</td> <td>Indicates that the distance weights process is NOT used for inter prediction.</td> </tr> <tr> <td>1h</td> <td></td> <td>Indicate that the distance weights process may be used for inter prediction.</td> </tr> </tbody> </table>	Value	Name	Description	0h	[Default]	Indicates that the distance weights process is NOT used for inter prediction.	1h		Indicate that the distance weights process may be used for inter prediction.
Value	Name	Description									
0h	[Default]	Indicates that the distance weights process is NOT used for inter prediction.									
1h		Indicate that the distance weights process may be used for inter prediction.									
20	Sequence Enable Masked Compound Flag	<p>Format: U1</p> <p>It specifies whether the Masked Compound coding tool is enabled for the video sequence, or not.</p> <p>It is the sequence level syntax element, enable_masked_compound.</p> <p>Valid only in Decoder Mode</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>[Default]</td> <td>Indicates that the mode info for inter blocks do NOT contain the block level syntax element compound_type and others.</td> </tr> <tr> <td>1h</td> <td></td> <td>Indicates the block level syntax elements: compound_type and others, are present in the bitstream.</td> </tr> </tbody> </table>	Value	Name	Description	0h	[Default]	Indicates that the mode info for inter blocks do NOT contain the block level syntax element compound_type and others.	1h		Indicates the block level syntax elements: compound_type and others, are present in the bitstream.
Value	Name	Description									
0h	[Default]	Indicates that the mode info for inter blocks do NOT contain the block level syntax element compound_type and others.									
1h		Indicates the block level syntax elements: compound_type and others, are present in the bitstream.									
19	Sequence Enable Inter-Intra Compound Flag	<p>Format: U1</p> <p>It specifies whether the Inter-Intra Compound coding tool is enabled for the video sequence, or not.</p> <p>It is the sequence level syntax element, enable_interintra_compound.</p> <p>Valid in Decoder Mode only</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>[Default]</td> <td>Indicates that the mode info for inter blocks do NOT contain the block level syntax element interintra and others.</td> </tr> <tr> <td>1h</td> <td></td> <td>Indicates that the block level syntax elements: interintra and others, are present in the bitstream.</td> </tr> </tbody> </table>	Value	Name	Description	0h	[Default]	Indicates that the mode info for inter blocks do NOT contain the block level syntax element interintra and others.	1h		Indicates that the block level syntax elements: interintra and others, are present in the bitstream.
Value	Name	Description									
0h	[Default]	Indicates that the mode info for inter blocks do NOT contain the block level syntax element interintra and others.									
1h		Indicates that the block level syntax elements: interintra and others, are present in the bitstream.									
18	Sequence Enable Dual_Filter Flag	<p>Format: U1</p> <p>It specifies whether the Motion Comp horizontal and vertical interpolation filters are specified independently in the bitstream at the block level (inter-coded block) or not, for the video sequence.</p> <p>Set to 0 ; indicate only ONE MC filter type is specified in the bitstream, which is then used in both directions . (Default)</p> <p>Set to 1 ; indicate the MC filter type may be specified independently in the horizontal and vertical directions.</p> <p>It is the sequence level syntax element, enable_dual_filter.</p> <p>Note : MC filter type can be [EIGHTTAP, EIGHTTAP_SMOOTH, EIGHTTAP_SHARP, BILINEAR, and SWITCHABLE].</p> <p>Must be set to 0 in encoder mode</p>									
17	Sequence Enable Intra Edge Filter Flag										

AVP_PIC_STATE			
	<table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>It specifies whether the Intra Edge Filtering process is enabled for the video sequence or not. Set to 0 ; indicate that the Intra Edge Filter tool is DISabled. (Default) Set to 1 ; indicate that the Intra Edge Filter tool is ENabled. It is the sequence level syntax element, enable_intra_edge_filter.</p>	Format:	U1
Format:	U1		
16	<p>Sequence Enable Filter_Intra Flag</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>It specifies if the Filter_Intra coding tool is enabled for the video sequence. Set to 0 ; indicate the block level syntax element: use_filter_intra, is NOT present in the bitstream. (Default) Set to 1 ; indicate the block level syntax element: use_filter_intra, is present in the bitstream. It is the sequence level syntax element, enable_filter_intra. Note : if both enable_filter_intra and use_filter_intra are set to 1, the corresponding block is coded with filter_intra Must be set to 0 in encoder mode</p>	Format:	U1
Format:	U1		
15:13	<p>Reserved (for the expansion of Sequence Order Hint Bits Minus1)</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
12:10	<p>Sequence Order Hint Bits Minus1</p> <table border="1"> <tr> <td>Format:</td> <td>U3</td> </tr> </table> <p>It specifies the number of bits to be read from the bitstream for the frame header syntax elements: order_hint and ref_order_hint[i=0 to 6]. It is in the range of [0..7] . It is the sequence level syntax element, order_hint_bits_minus1, which is only present in the bitstream if the sequence level syntax element enable_order_hint_flag is set to 1. It is defaulted to 0, if not present. It can be used to generate the bitmask in computing the relative distance between two order hints. Note: it is also used for other purposes, as detail in the bitfield Sequence Enable Order Hint Flag.</p>	Format:	U3
Format:	U3		
9	<p>Sequence Enable Order Hint Flag</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>Set to 1 ;enables the use of Order Hint in the decoding process of all INTER frames in the video sequence. Set to 0 ; disables the use of Order Hint. (Default) It is the sequence level syntax element, enable_order_hint. It controls 1) the bitstream reading of the syntax elements: enable_jnt_comp, enable_ref_frame_mvs, and order_hint_bits_minus1 in the sequence header. 2) the bitstream reading of the syntax elements: ref_order_hint[i=0 to 6] and frame_refs_short_signaling in the frame header. 3) the setting of reference frames in the frame header for the decoding of the current frame. Note : these 2 sequence level syntax elements: enable_order_hint flag and the 3-bits</p>	Format:	U1
Format:	U1		

AVP_PIC_STATE			
	<p>order_hint_bits_minus1 are used in :</p> <ol style="list-style-type: none"> 1) skip_mode derivation, 2) compound prediction temporal weighing factor derivation, 3) motion field estimation process and MV projection, 4) motion_field MV storage setting, 5) reference frame bias derivation, 6) forward, second_forward, and backward reference frame selection, 7) and forward, second_forward and backward reference frame order hint setting. 		
8:7	<p>Sequence Superblock Size Used</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It specifies one of the two possible Superblock sizes that is used to code the video sequence. Set to 0, if the SuperBlock size is 64x64 pixels. (Default) Set to 1, if the SuperBlock size is 128x128 pixels. Value 2-3 are reserved. It is the sequence level syntax element, use_128x128_superblock; and is also named as sb_size. Supports SB size 64x64 only in encoder mode</p>	Format:	U2
Format:	U2		
6	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
5	<p>Reserved (for expansion of Sequence Pixel Bit-Depth Idc)</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
4:3	<p>Sequence Pixel Bit-Depth Idc</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It specifies the pixel bit depth for all frames in the video sequence being decoded. [4:3] = 00 ; specifies 8-bit per pixel (bpp). (Default) [4:3] = 01 ; specifies 10-bit per pixel. [4:3] = 10 ; specifies 12-bit per pixel It is a sequence-level parameter derived from the sequence header syntax elements:seq_profile,high_bitdepth and twelve_bit. Note : Only Bit-Depth = 8 and 10 are allowed for this generation of AVP. Note : refer to the description under Sequence Chroma Sampling Format for detail on allowed bit depth for each profile.</p>	Format:	U2
Format:	U2		
2	<p>Reserved (for expansion of Chroma SubSampling Format)</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
1:0	<p>Sequence Chroma SubSampling Format</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It specifies the chroma subsampling format for all frames in the video sequence being decoded. [1:0] = 00 ; stands for Monochrome 4:0:0, no Chroma planes at all, but [subsampling_x and subsampling_y] is defaulted to [1, 1], as only Profile 0 can support monochrome video coding. [1:0] = 01 ; stands for 4:2:0, with[subsampling_x and subsampling_y] defining as[1, 1]. It is</p>	Format:	U2
Format:	U2		

AVP_PIC_STATE

supported in all profiles (seq_profile=0, 1, 2 - syntax element in the sequence header)
 [1:0] = 10 ; stands for 4:2:2, with[subsampling_x and subsampling_y] defining as[1, 0]. It is supported only in seq_profile=2.
 [1:0] = 11 ; stands for 4:4:4 with[subsampling_x and subsampling_y] defining as[0, 0]. It is supported in both seq_profile=1 and 2.
 It is a sequence-level parameter derived from the sequence header syntax elements: seq_profile, subsampling_x, subsampling_y, monochrome, high_bitdepth and twelve_bit. Default is 1, i.e. 4:2:0..

Note : AV1 supports 3 profiles:
 seq_profile Bit_depth Chroma Subsampling
 0 (Main Profile) 8 / 10 YUV 4:2:0 and 4:0:0
 1 (High Profile) 8 / 10 YUV 4:4:4 (4:0:0 is not allowed)
 2 (Pro Profile) 8 / 10 /12 YUV 4:2:2 AND
 12 YUV 4:2:0/4:4:4/4:0:0

Note : for AV1 decoder:

- A profile 0 compliant decoder must be able to decode all bitstreams labeled profile 0
- A profile 1 compliant decoder must be able to decode all bitstreams labeled profile 0 or 1
- A profile 2 compliant decoder must be able to decode all bitstreams labeled profile 0, 1, or 2

"01" -- Chroma Sampling 4:2:0 is supported.

Value	Name	Description
1h	4:2:0	Chroma Sampling 4:2:0

3

31

Reserved (for future expansion of Primary Reference Frame Idx)

Format:	MBZ
---------	-----

30:28

Primary Reference Frame Idx

Format:	U3
---------	----

It specifies which one of the 7 possible reference frames [reference frame ID0 to 6] contains the frame context(CDF table set for all syntax elements) and other state that should be loaded at the start of the frame.

The normal range of values for Primary Reference Frame is [0 to 6]. The value of 7 (defined as PRIMARY_REF_NONE) is used to signal when there is no primary reference frame.

It is the frame level syntax element, primary_ref_frame, which is present in the bitstream when (! FramelsIntra && ! error_resilient_mode). If it is not present in the bitstream, it is defaulted to PRIMARY_REF_NONE (=7).

There is no primary reference frame:

1) if (FramelsIntra || error_resilient_mode). Hence only INTER Frame and SWITCH Frame can have primary reference frame ID specified in the bitstream when not in error resilient mode.

2) Or when it receives a value of 7from the bitstream.

Note : load_cdfs(ref_frame_idx[primary_ref_frame])is a function call that indicates that the frame context (CDF table set)is loaded from the frame context attached to one of the 8 possible reference frames in the DPB.

AVP_PIC_STATE									
	<p>Note : load_previous() is a function call that indicates that information from a previous frame may be loaded for use in decoding the current frame.</p> <p>Note : if (primary_ref_frame == PRIMARY_REF_NONE), it is required to set segmentation_update_map = 1 , segmentation_temporal_update = 0 , and segmentation_update_data = 1.</p> <p>Note : If primary_ref_frame is set to PRIMARY_REF_NONE, it is a requirement of bitstream conformance that loop_filter_delta_update is equal to 1.</p>								
27:24	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
23	<p>Allow IntraBC Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U1</td> </tr> </table> <p>It specifies whether intra block copy prediction mode is allowed to be used in the current frame or not.</p> <p>Set to 0 to disallow intra block copy prediction mode. (Default)</p> <p>Set to 1 to allow intra block copy prediction mode.</p> <p>It is the frame level syntax element, allow_intrabc. It is present in the bitstream, only if the sequence syntax element allow_screen_content_tools is set to 1 AND there is no super-resolution frame width scaling (but super-resolution can still be enabled). When it is not present in the bitstream, it is defaulted to 0.</p> <p>Note : intra block copy is only allowed in KEY Frame and INTRA-ONLY Non-Key Frame. For all other frame types (INTER Frame and SWITCH Frame), this flag is set to 0.</p> <p>Note : when this flag is set to 1, all post in loop filters (deblocker, CDEF, Super-Resolution and Loop Restoration) are all disabled.</p> <p>Note : when this flag is set to 1, force_integer_mv is set to 1 in KEY Frame and INTRA-ONLY Non-Key Frame.</p> <p>Valid only in Decoder Mode</p>	Format:	U1						
Format:	U1								
22	<p>Error Resilient Mode Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U1</td> </tr> </table> <p>It specifies whether all syntax decoding of the current frame is independent of the previous frames, or not.</p> <p>Set to 0 to disable error resilient mode</p> <p>Set to 1 to enable error resilient mode (for independent syntax decoding)</p> <p>It is the frame-level syntax element, error_resilient_mode. Default is 0.</p> <p>It is read from the bitstream for all frame types (KEY Frame, INTRA-ONLY Frame and INTER Frame), except when frame_type is set to SWITCH_FRAME, in which it is forced to 1 instead of reading from the bitstream.</p> <p>When error resilient mode is set to 1 (active), Refresh Frame Context is set to 0. When error resilient is set to 0, Refresh Frame Context is read from the bit stream.</p> <p>Valid only in Decoder Mode</p> <table border="1" style="width: 100%; margin-top: 10px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Enable</td> </tr> </tbody> </table>	Format:	U1	Value	Name	0	Disable	1	Enable
Format:	U1								
Value	Name								
0	Disable								
1	Enable								

AVP_PIC_STATE			
21:20	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
19	<p>IntraOnly Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U1</td> </tr> </table> <p>It specifies if the current frame being decoded is a KEY Frame or a INTRA-ONLY Non-Key Frame. It is a derived parameter from the frame level syntax element, frame_type, and is also named as FrameIntra.</p> <p>IntraOnly Flag = (frame_type == INTRA_ONLY_FRAME) (frame_type == KEY_FRAME).</p>	Format:	U1
Format:	U1		
18	<p>Reserved (for the expansion of Frame Type)</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
17:16	<p>Frame Type</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U2</td> </tr> </table> <p>Specifies one of the four possible AV1 frame types.</p> <p>[17:16] = [00] ; specifies KEY Frame. (Default)</p> <p>[17:16] = [01] ; specifies INTER Non-Key Frame.</p> <p>[17:16] = [10] ; specifies INTRA-ONLY Non-Key Frame.</p> <p>[17:16] = [11] ; specifies SWITCH Non-Key Frame (or called S-Frame, is a different type of INTER Frame).</p> <p>It is the frame level syntax element, frame_type.</p> <p>Note: Encoder does not support INTRA-ONLY and S-Frame</p>	Format:	U2
Format:	U2		
15:14	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
13:9	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
8:7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
6	<p>Frame Level Loop Restoration Filter Enable Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U1</td> </tr> </table> <p>It specifies whether the Post In-Loop Loop Restoration Filter tool is enabled for the current frame, or not.</p> <p>Set to 0 ; indicate that the Loop Restoration Filter is DISabled for the current frame.(Default)</p> <p>Set to 1 ; indicate that the Loop Restoration Filter is ENabled for the current frame.</p> <p>Frame Level Loop Restoration Filter Enable Flag is an intel derived parameter, and is set to 1 when Luma frame_restoration_type != RESTORE_NONE Chroma Cb frame_restoration_type != RESTORE_NONE Chroma Cr frame_restoration_type != RESTORE_NONE;</p> <p>It is derived from 1) the sequence level syntax element, enable_restoration, 2) the frame level derived coding parameters : frame level all_lossless and allow_intrabc, 3) and also the frame level syntax for loop restoration filter type of each color component.</p> <p>Although Use Loop Restoration Filter Flag = ! (AllLossless allow_intrabc !enable_restoration) can be used to signal the disabling of the loop restoration filter for the current frame, it is still</p>	Format:	U1
Format:	U1		

AVP_PIC_STATE

	<p>possible at the frame level to read in the loop restoration filter type syntax elements that indicate the loop restoration filter is not enabled. To simplify hardware, this field is used to give HW the derived final decision if the loop restoration filter is enabled or not.</p> <p>When Large Scale Tile Enable flag is set to 1, this Loop Restoration Filterflag must set to 0, to disable the loop restoration filtering completely.</p> <p>Note : When individual frames in a video sequence are coded in Alllossless or with intraBC enabled, all post in-loop filtering processes (deblocker, CDEF, horizontal super-res, and loop restoration) should be disabled.</p> <p>Note : When Loop Restoration filter is disabled, all its filter types should be programmed to RESTORE_NONE.</p>		
5	<p>Use Super-Res Flag</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%; text-align: center;">U1</td> </tr> </table> <p>It specifies whether the Post In-Loop Horizontal-Only Super-Resolution tool is enabled for the current frame being decoded, or not.</p> <p>Set to 0 ; indicate that the Super-Res Frame Upscaling process is NOT performed. This sequence level setting cannot be overridden. (Default)</p> <p>Set to 1 ; indicate that the Super-Res Frame Upscaling process is to be performed. But this sequence level setting can be overridden when in frame coded lossless or when intraBC is enabled.</p> <p>It is derived from the sequence level syntax element, enable_superres, from the frame level syntax element of the same name: use_superres, and from the frame level derived coding parameters : frame coded_lossless and allow_intrabc.</p> <p>If the frame level syntax element of the same name, use_superres is not present in the bitstream, it is defaulted to 0.</p> <p>When Large Scale Tile Enable flag is set to 1, this Use Super-Res flag must set to 0, to disable super-resolution coding method.</p> <p>Note : it is legal to set the sequence level syntax element: enable_superresequel to 1, even when use_superres is set to 0 (i.e. no superres is performed)on all frames in the coded video sequence.</p> <p>Note : When individual frames in a video sequence are coded in coded_lossless or with intraBC enabled, all post in-loop filtering processes (deblocker, CDEF, horizontal super-res, and loop restoration) should be disabled.</p> <p>Note : When Super-res is disabled, the Denom of the scaling factor should be set to 8 (to have the same value as the default NUMERATOR).</p>	Format:	U1
Format:	U1		
4	<p>Use CDEF Filter Flag</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%; text-align: center;">U1</td> </tr> </table> <p>It specifies whether the Post In-Loop CDEF Filter tool is enabled for the current frame being decoded, or not.</p> <p>Set to 0 ; indicate that the CDEF Filter is DISabled. This sequence level setting cannot be overridden. (Default)</p> <p>Set to 1 ; indicate that the CDEF Filter is ENabled. But this sequence level setting can be overridden when in frame-level coded lossless or when intraBC is enabled.</p> <p>It is derived from the sequence level syntax element of the same name, enable_cdef and the frame level derived coding parameters : coded_lossless and allow_intrabc.</p>	Format:	U1
Format:	U1		

AVP_PIC_STATE			
	<p>Use CDEF filter Flag = ! (CodedLossless allow_intraBC !enable_cdef).</p> <p>When Large Scale Tile Enable flag is set to 1, this Use CDEF Filter flag must set to 0, to disable CDEF Filtering operation completely.</p> <p>Note : it is legal to set enable_cdef equal to 1 even when cdef filtering is not used on any frame in the coded video sequence.</p> <p>Note : When individual frames in a video sequence are coded in lossless or with intraBC enabled, all post in-loop filtering processes (deblocker, CDEF, horizontal super-res, and loop restoration) should be disabled.</p> <p>Note : When CDEF filter is disabled, all its filter parameters should be reset to 0.</p>		
3	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
2	<p>Allow Warped Motion Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U1</td> </tr> </table> <p>Set to 0 ; indicates that the block level syntax element: motion_mode is NOT present in the bitstream. Hence, the block cannot signal for LOCALWARP coding.</p> <p>Set to 1 ; indicates that the block level syntax element: motion_mode is present in the bitstream. It is the frame level syntax element, allow_warp_motion, which is present in the bitstream only if this condition is met: (! FrameIntra && ! error_resilient_mode && enable_warped_motion). If it is not present in the bitstream, it is defaulted to 0.</p> <p>Note : motion_mode can take on a value of [SIMPLE, OBMC, or LOCALWARP]. LOCALWARP not supported in encoder mode</p>	Format:	U1
Format:	U1		
1	<p>Force Integer MV Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U1</td> </tr> </table> <p>Set to 0 ; indicates that motion vectors used in the Motion Comp can contain fractional bits (sub-pel precision). (Default)</p> <p>Set to 1 ; indicates that motion vectors used in the Motion Comp is always integers (NO sub-pel precision for Luma only).</p> <p>It is derived from these sequence level syntax elements: seq_choose_screen_content_tools and seq_force_screen_content_tools, and from the frame level syntax element : allow_screen_content_tools.</p> <p>if (allow_screen_content_tools = =1) AND (seq_force_integer_mv = = 2), read the value of force_integer_mv flag from the bitstream (in the frame header).</p> <p>If(allow_screen_content_tools = =1) AND (seq_force_integer_mv != 2) , force_integer_mv flag is set to the value of seq_force_integer_mv (which is read from the bitstream in the sequence header).</p> <p>if(allow_screen_content_tools = =0) , force_integer_mv is set to 0.</p> <p>Note : for 4:2:0 and 4:2:2, the chroma subsampling factors (subsampling_x and subsampling_y) are also applied to the Luma MVs. Hence, integer Luma MVs will still give half-pel precision Chroma MVs (i.e. with fractional bit).</p> <p>Note : if intraBC is enabled, force_integer_mv is always set to 1 in the intra frames (KEY Frame or Intra-Only Non-Key Frame).</p> <p>Note : when force_integer_mv is set to 1, some fractional bits are still read for the translation components. However, these fractional bits will be discarded during the Setup Zero MV process.</p>	Format:	U1
Format:	U1		

AVP_PIC_STATE					
	Valid only in decoder Mode				
0	<p>Allow Screen Content Tools Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Set to 0 ; indicates that the two block level screen content coding tools (palette prediction coding and intraBC coding). (Default)</p> <p>Set to 1 ; indicates that the block level syntax element: motion_mode is present in the bitstream. It is derived from thesequence level syntax elements: seq_choose_screen_content_tools andseq_force_screen_content_tools, and from the frame level syntax element of the same name: allow_screen_content_tools.</p> <p>ifseq_choose_screen_content_tools = = 1, read the value of allow_screen_ccontent_tools from the bitstream (in the frame header).</p> <p>Ifseq_choose_screen_content_tools = = 0, allow_screen_content_tools is set to the value of seq_force_screen_content_tools (which is read from the bitstream in the sequence header).</p> <p>Note : at the frame header, allow_screen_content_tools flag controls the setting of intraBC and integer_mv coding tools, but at the block level, it controls the use of the palette prediction coding mode.</p> <p>Valid in Decoder Mode only</p>	Format:	U1		
Format:	U1				
4	31	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
	Format:	MBZ			
	30:24	<p>Y_dc_delta_q</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S6</td> </tr> </table> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table> <p>2's complement sign number of range -63 to +63.</p>	Format:	S6	Programming Notes
	Format:	S6			
	Programming Notes				
23:16	<p>Base Qindex</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It can take on value range from 0 to 255.</p> <p>This is the same as the Y_ac_q, becasue y_ac_delta_q is always set to 0.</p>	Format:	U8		
Format:	U8				
15:14	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ				
13	<p>Segment ID Buffer Stream-Out Enable Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>It indicates if the processing of the current frame requires to write out the segment IDs at the block level to a segment ID Buffer.</p> <p>It is set to 1, if stream-outis going to happen.</p> <p>It is set to 0, if stream-outis not going to happen at all.</p> <p>It is an intel derived frame level parameters based on only frame level syntax and other derived parameters. Its meaning is defined as follows:</p> <p>if(SegmentIdStreamOutFlag)</p>	Format:	U1		
Format:	U1				

AVP_PIC_STATE

		<pre> { HW needs to write out segment ID to Segment ID Write Buffer; } else { HW does not need to write out segment ID; } Driver is responsible to set this Segment ID Buffer Stream-Out Enable Flag based on the following conditions (using Driver variable names) : Frame Level derived parameter : (seg->enable == 1) && Frame Level SE : seg->update_map == 1 Valid in Decoder only Mode </pre>		
12	Segment ID Buffer Stream-In Enable Flag	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>It indicates if the processing of the current frame requires to read in the segment IDs at the block level from a segment ID Buffer.</p> <p>It is set to 1, if stream-in is going to happen.</p> <p>It is set to 0, if stream-in is not going to happen at all.</p> <p>It is an intel derived frame level parameters based on only frame level syntax and other derived parameters. Its meaning is defined as follows:</p> <p>Valid in Decoder only Mode</p> <pre> if (SegmentMapsZeroFlag) { Segment ID is all 0s. Driver will programSegmentIdStreamInFlagto 0. (HW checks on the SegmentMapsZeroFlag=1, and set all temporal segment ID to 0) } else if (SegmentIdStreamInFlag) { Segment ID is streamed in from Segment ID read buffer; } else { No Segment ID stream-in } </pre> <p>Driver is responsible to set this Segment ID Buffer Stream-InEnable Flag based on the following conditions (using Driver variable names) :</p> <pre> Frame Level derived parameter : (seg->enable == 1) && Frame Level SE : (primary_ref_frame != PRIMARY_REF_NONE) && Frame Level derived parameters : ((seg->update_map == seg->temporal_update == 1) (seg->update_map == seg->temporal_update == 0)) && ((DPB[current_frame].resolution == DPB[primary_ref_frame].resolution) && (DPB[primary_ref_frame].seg_enable) && (DPB[primary_ref_frame].segment_ID_buffer != NULL)) </pre>	Format:	U1
Format:	U1			

AVP_PIC_STATE

11	Segment Map Is Zero Flag	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>In a video sequence or within a GOP, some frames can have segmentation disabled. Their corresponding segment map is defaulted to have its content set to all 0. If later, this all zero segment map will be referenced by HW for temporal segment map prediction, it will be a waste of bandwidth to read in zeros.</p> <p>Segment Map Is Zero flag is an intel added parameter at frame level.</p> <p>If it is set to 1, it tells HW that the segment map is containing all zero. AVP HW will check this bit together with if segment map streamin is enabled, then HW will not actually read from the segment map buffer, but will internally generate the read of 0 instead, to save bandwidth.</p> <p>Driver needs to keep track of which reference frame(s) (max 7) have segmentation disabled for decoding the current frame, and set this flag accordingly.</p> <p>Driver is responsible to set this Segment Map Is Zero Flag based on the following conditions (using Driver variable names) :</p> <p>Frame Level derived parameter : (seg->enable == 1) && Frame Level SE : (primary_ref_frame != PRIMARY_REF_NONE) && Frame Level derived parameters : ((seg->update_map == seg->temporal_update == 1) (seg->update_map == seg->temporal_update == 0)) && ! [((DPB[current_frame].resolution == DPB[primary_ref_frame].resolution) && (DPB[primary_ref_frame].seg_enable) && (DPB[primary_ref_frame].segment_ID_buffer != NULL))]</p>	Format:	U1						
Format:	U1									
10	Frame Coded Lossless Mode	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>This bitSet to indicate lossless coding mode at frame level.</p> <p>Frame Coded Lossless Mode is set to 1, if all active segment's segment lossless flag are set to 1. The equation for deriving coded lossless mode is presented in the AVP_SEGMENT_STATE Command.</p> <p>AllLossless = CodedLossless && (FrameWidth == UpscaledWidth). The second condition in this equation is equivalent to having Super-res NOT enabled.</p> <p>Only CodedLossless flag is sent to HW. AllLossless flag is not.</p> <p>CodedLossless directly control the enabling/disabling of deblocker, CDEF in-loop filters. But AllLossless is used to control the enabling/disabling of Loop Restoration filter. Hence, when super-res is ON, Loop Restoration filter can still be ON/OFF, regardless of CodedLossless.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal Mode</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Coded Lossless Mode</td> </tr> </tbody> </table>	Format:	U1	Value	Name	0	Normal Mode	1	Coded Lossless Mode
Format:	U1									
Value	Name									
0	Normal Mode									
1	Coded Lossless Mode									
9:8	Delta Q RES	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>if delta_q_present_flag = 1, 2 bits from bitstream are read. Delta_q_res = 0, 1, 2 and 3. and the mulitple factor= 1 « (delta_q_res) = [1, 2, 4 or 8].</p>	Format:	U2						
Format:	U2									

AVP_PIC_STATE									
	Here, only the 2 bits are programmed to the HW.								
7	<p>Delta Q Present Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>Enable</td> </tr> </table> <p>if delta_q_present_flag = 1, delta_q_res is present in the bitstream - to specify SB level delta q. In encoder mode, this flag must be set to 1 for BRC purpose so the QP can be changed at SB level. However, it can be set to 0 for CQP across frame</p>	Format:	Enable						
Format:	Enable								
6:4	<p>Last Active Segment ID</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U3</td> </tr> </table> <p>It specifies the highest numbered segment id that has some enabled feature. This is used when decoding the segment id to only decode choices corresponding to used segments.</p>	Format:	U3						
Format:	U3								
3	<p>Pre-Skip Segment ID Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U1</td> </tr> </table> <p>Set to 1 indicates that the segment id will be read before the skip syntax element. SegIdPreSkip equal to 0 indicates that the skip syntax element will be read first.</p>	Format:	U1						
Format:	U1								
2	<p>Segmentation Temporal Update Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U1</td> </tr> </table> <p>Indicates whether segID is decoding from bitstream or predicted from previous frame. In encoder Mode it should use either from previous frame or streamIn</p> <table border="1" style="width: 100%; margin-top: 10px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Decode segID from bitstream</td> </tr> <tr> <td>1h</td> <td>Get segID either from bitstream or from previous frame</td> </tr> </tbody> </table> <p style="text-align: center; margin-top: 10px;">Programming Notes</p> <p>Decoder Only: For KEY_FRAME or INTRA_ONLY frame [OR ERROR_RESILIENCE], this bit should be set to "0". [Temporary add in the condition ERROR_RESILIENCE, to be removed later] Note: Driver should override this flag to "0" in KEY_FRAME or INTRA_ONLY frame even if this bit decoded from bitstream is different. This is for hardware optimization. This override does not affect bitstream decoding other than uncompressed header.</p>	Format:	U1	Value	Name	0h	Decode segID from bitstream	1h	Get segID either from bitstream or from previous frame
Format:	U1								
Value	Name								
0h	Decode segID from bitstream								
1h	Get segID either from bitstream or from previous frame								
1	<p>Segmentation Update Map Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U1</td> </tr> </table> <p>Set to 0 means that the segmentation map from the previous frame is used, and the current frame decode is not changing the segmentation map. (Default) Set to 1 means that the segmentation map is updated during the decoding of the current frame. Hence SegmentIDs of current frame are streamout to a write only surface, which will be used as</p>	Format:	U1						
Format:	U1								

AVP_PIC_STATE											
	<p>the segmentation map for a future frame(s). It is the frame level syntax element, segmentation_update_map. It is present in the bitstream only if, the frame level syntax elements: segmentation is enabled AND primary_ref_frame is NOT PRIMARY_REF_NONE. If primary_ref_frame is set to PRIMARY_REF_NONE, segmentation_update_map is always set to 1. Note : the segmentation map that has streamout (surface buffer) is attached to the current frame, after it has become a reference frame in the Display Buffer (DPB). Note : HW needs to detect one of these 3 conditions (the current frame is in error resilient mode OR the current frame type is KEY Frame or INTRA-ONLY Frame), and if TRUE, then HW will not read the segment map for decoding the current frame, all segment ID is considered as 0. But during the decoding of the current frame, the segment map can still be modified is segmentation map update flag is ON.</p>										
0	<p>Segmentation Enable Flag</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> <tr> <td colspan="2">Indicate if segmentation is enabled or not</td> </tr> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Name</th> </tr> <tr> <td>0h</td> <td>All blocks are implied to belong to segment 0</td> </tr> <tr> <td>1h</td> <td>SegID determination depends on segmentation_update_map setting</td> </tr> </table>	Format:	U1	Indicate if segmentation is enabled or not		Value	Name	0h	All blocks are implied to belong to segment 0	1h	SegID determination depends on segmentation_update_map setting
Format:	U1										
Indicate if segmentation is enabled or not											
Value	Name										
0h	All blocks are implied to belong to segment 0										
1h	SegID determination depends on segmentation_update_map setting										
5	31	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
	30:24	<p>V_ac_delta_q</p> <table border="1"> <tr> <td>Format:</td> <td>S6</td> </tr> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">2's complement sign number of range -63 to +63.</td> </tr> </table>	Format:	S6	Programming Notes		2's complement sign number of range -63 to +63.				
	Format:	S6									
	Programming Notes										
	2's complement sign number of range -63 to +63.										
23	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
22:16	<p>V_dc_delta_q</p> <table border="1"> <tr> <td>Format:</td> <td>S6</td> </tr> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">2's complement sign number of range -63 to +63.</td> </tr> </table>	Format:	S6	Programming Notes		2's complement sign number of range -63 to +63.					
Format:	S6										
Programming Notes											
2's complement sign number of range -63 to +63.											
15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
14:8	<p>U_ac_delta_q</p> <table border="1"> <tr> <td>Format:</td> <td>S6</td> </tr> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">2's complement sign number of range -63 to +63.</td> </tr> </table>	Format:	S6	Programming Notes		2's complement sign number of range -63 to +63.					
Format:	S6										
Programming Notes											
2's complement sign number of range -63 to +63.											

AVP_PIC_STATE			
	7	Reserved	Format: MBZ
	6:0	U_dc_delta_q	Format: S6
		Programming Notes	
		2's complement sign number of range -63 to +63.	
6	31:28	Reserved	Format: MBZ
	27:24	Reserved (for future expansion of Frame Order Hint)	Format: MBZ
	23:16	Current Frame Order Hint	Format: U8
		<p>It specifies "OrderHintBits" least significant bits of the expected output order for the current frame being decoded.</p> <p>It is the frame level syntax element, order_hint. Default value is 0.</p> <p>Order_hint is a variable bit length syntax element; its bit length is in the range of OrderHintBits=[1..8]. Hence, maximum order_hint can have a value of 255.</p> <p>Note: There is no requirement that OrderHint should reflect the true output order.</p>	
15:8	Reference Frame Sign Bias [i=0 to 7]	Format: U8	
		<p>This is a bit array that specifies the Reference Frame Sign Bias for Reference 0 to 7 (only 1 to 7 are valid, OR 0 to 6 ???)</p> <p>Reference Picture 0 is INTRA frame and there is no sign bias for it.</p> <p>Bit 9 : Last Frame (Reference Picture 1)</p> <p>Bit 10: Last2 Frame (Reference Picture 2)</p> <p>Bit 11: Last3 Frame (Reference Picture 3)</p> <p>Bit 12: GoldenFrame (Reference Picture 4)</p> <p>Bit 13: Bwdref Frame (Reference Picture 5)</p> <p>Bit 14: Altref2 Frame (Reference Picture 6)</p> <p>Bit 15: Altref Frame (Reference Picture 7)</p> <p>It is a frame-level derived parameter for each reference frame, RefFrameSignBias[i=0 to 7]. 1-bit per reference frame. It is derived from the frame level parameters order hint and ref order hints. If the sequence syntax element enable_order_hint is set to 0, RefFrameSignBias[i=0 to 7] are all set to 0.</p>	
	7	Use Reference Frame MV Set Flag	Format: U1
		<p>Set to 0, specifies that motion vector information from a previous frame (stored inside the temporal MV buffer) CANNOT be used when decoding the current frame. (Default)</p> <p>Set to 1, specifies that motion vector information from a previous frame can be used when</p>	

AVP_PIC_STATE															
	<p>decoding the current frame.</p> <p>It is the frame level syntax element, use_ref_frame_mvs, which is present in the bitstream only if the sequence level syntax elements: enable_order_hint and enable_ref_frame_mvs are both set to 1 and the frame level syntax element: error_resilient_mode is set to 0. If it is not present in the bitstream, it is defaulted to 0.</p>														
6	<p>Motion Mode Switchable Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Set to 0, specifies only SIMPLE motion mode can be used. (Default) Set to 1, specifies the motion mode being used is determined at the block level. It is the frame level syntax element, is_motion_mode_switchable. It is present only in INTER Frame or SWITCH Frame. For all other frame types (KEY Frame and INTRA-ONLY Frame), it is defaulted to 0. Note :Motion Mode for motion comp can be SIMPLE, OBMC or LOCALWARP.</p>	Format:	U1												
Format:	U1														
5	<p>Reserved (for future expansion of Mcomp Filter Type)</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ												
Format:	MBZ														
4:2	<p>Mcomp Filter Type</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U3</td> </tr> </table> <p>It specifies which Motion Compensation Filter type is to be used for the current frame. It is a frame-level derived parameters. It is derived from the frame level syntax elements (is_filter_switchable flag and the 2-bits interpolation_filter). Default is 0 (i.e. use the eight-tap basic filter).</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 70%;">Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Eight-tap</td> </tr> <tr> <td>1h</td> <td>Eight-tap-Smooth</td> </tr> <tr> <td>2h</td> <td>Eight-tap-Sharp</td> </tr> <tr> <td>3h</td> <td>Bilinear</td> </tr> <tr> <td>4h</td> <td>Switchable</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, setting Switchable results in Eight-tap filter acrosss the whold frame.</p>	Format:	U3	Value	Name	0h	Eight-tap	1h	Eight-tap-Smooth	2h	Eight-tap-Sharp	3h	Bilinear	4h	Switchable
Format:	U3														
Value	Name														
0h	Eight-tap														
1h	Eight-tap-Smooth														
2h	Eight-tap-Sharp														
3h	Bilinear														
4h	Switchable														
1	<p>Frame Level Reference Mode Select</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Set to 0 specifies that all inter blocks in the current frame will use single prediction (SINGLE_REFERENCE). Default is 0, use SINGLE_REFERENCE. Set to 1 specifies that the frame level prediction mode for inter blocks is REFERENCE_MODE_SELECT, which will cause reading the syntax element comp_mode at PartU level to specify whether to use single or compound reference prediction for that partU. It is the frame level syntax element, reference_select. If it is NOT present in the bitstream, it is defaulted to 0.</p>	Format:	U1												
Format:	U1														

AVP_PIC_STATE			
0	<p>Allow High Precision MV</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>It specifies whether the high precision MV mode is used for the Luma Motion Vector prediction or not. Set to 0, specifies that motion vectors are in quarter-pel precision. (Default, the normal mode) Set to 1, specifies that motion vectors are in eighth-pel precision. It is the frame-level syntax element, allow_high_precision_mv. It is present in the bitstream, only if the frame level parameter force_integer_mv is set to 0. If it is not present in the bitstream, it is default to 0.</p>	Format:	U1
Format:	U1		
7	<p>Reference Frame Side [i=0 to 7]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This is a bit array that specifies the Reference Frame Side for Reference 0 to 7. Bit 24: Intra Frame (Reference Picture 0) Bit 25: Last Frame (Reference Picture 1) Bit 26: Last2 Frame (Reference Picture 2) Bit 27: Last3 Frame (Reference Picture 3) Bit 28: GoldenFrame (Reference Picture 4) Bit 29: Bwdref Frame (Reference Picture 5) Bit 30: Altref2 Frame (Reference Picture 6) Bit 31: Altref Frame (Reference Picture 7) It is a intel frame-level derived parameter for each reference frame, For each reference frame the corresponding bit is set to 0 when the corresponding bit in ref_frame_side is 0, else the bit will be set to 1. Each individual bit (i=0 to 7) of the original reference frame side parameter can take on a value of -1, 0, or 1. But intel version of the same parameter can only take on a value of 0 or 1. Both the original value of -1 and 1 is mapped to 1 here, and the original value of 0 remains mapped to 0. Default all 8-bits are set to 0.</p>	Format:	U8
Format:	U8		
23:13	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
12	<p>Reserved (for future expansion of Skip Mode Frame[1])</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
11:9	<p>Skip Mode Frame [1]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U3</td> </tr> </table> <p>It specifies the reference frames to use for compound prediction when skip_mode is set to 1. It is the frame level derived parameter SkipModeFrame[1]. Skip mode tries to use the closest forward (past) and backward (future) references (as measured by values in the RefOrderHint array). If no backward reference is found, then the second closest forward reference is used. If no forward reference is found, then skip mode is disabled.</p>	Format:	U3
Format:	U3		
8	<p>Reserved (for future expansion of Skip Mode Frame[0])</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		

AVP_PIC_STATE			
7:5	<p>Skip Mode Frame [0]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U3</td> </tr> </table> <p>It specifies the reference frames to use for compound prediction when skip_mode is set to 1. It is the frame level derived parameter SkipModeFrame[0]. Skip mode tries to use the closest forward (past) and backward (future) references (as measured by values in the RefOrderHint array). If no backward reference is found, then the second closest forward reference is used. If no forward reference is found, then skip mode is disabled.</p>	Format:	U3
Format:	U3		
4	<p>Skip Mode Present Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Set to 0 specifies that skip_mode will not be used for this frame. Default is 0, no PartU level skip flag. Set to 1 specifies that the syntax element skip_mode will be coded in the bitstream at the PartU level. It is the frame-level syntax element skip_mode_present. It is present in the bitstream based on an algorithm using RefOrderHint. If it is not present in the bitstream, it is defaulted to 0.</p>	Format:	U1
Format:	U1		
3	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
2:1	<p>Frame Transform Mode</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It specifies the Luma-only transform size to be used for the entire current frame to be decoded. 1) tx_mode=0 for ONLY_4X4, if the current frame is coded with frame coded lossless and only Hadamard 4x4 transform is being used for the whole frame. This is also applied to Chroma planes as well. 2) tx_mode=1 for TX_MODE_LARGEST, the inverse transform will use the largest transform size that fits inside the block. 3) tx_mode=2 for TX_MODE_SELECT, the choice of transform size is specified explicitly for each block. It is the frame-level derived parameters, TxMode. It is derived from the frame-level syntax element, tx_mode_select and the frame level derived parameter Frame Coded Lossless. Default is TxMode=1. Chroma Tx Mode is no longer derived from that of Luma Tx size. Chroma Tx Mode has no explicit setting in the bitstream, and Chroma Tx Mode can be viewed as always equivalent to TX_MODE_LARGEST for the two Chroma planes.</p>	Format:	U2
Format:	U2		
0	<p>Reduced Tx Set Used</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>set to 1 to allow the use of a reduced tx set. set to 0 to allow the full tx set to be used for each tx type. It is the frame level syntax element, reduced_tx_set.</p>	Format:	U1
Format:	U1		

AVP_PIC_STATE			
8	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">31:24</td> <td> Frame Level Global Motion Invalid Flags Format: U8 </td> </tr> </table> <p>It indicates to the AV1 HW pipeline (at block level processing) that the result of parsing the frame level global motion parameters of each reference frame is invalid or not. Each bit represents the invalid flag of a reference frame : it is set to 1, if the global motion parameters for the corresponding reference frame is invalid. Otherwise it is set to 0 for valid (default). Frame Level Global Motion Invalid Flags[31] indicates the validity of reference frame ALTREF. ... Frame Level Global Motion Invalid Flags[25] indicates the validity of reference frame LAST. Frame Level Global Motion Invalid Flags[24] indicates the validity of reference frame INTRA (this bit is reserved and not being used by HW)... This is an intel defined parameter to give HW a hint of frame header parsing result that can simplify the HW design. It takes on the same value as of the global_motion[ALTRF...LAST].invalid flag, which is set insideread_global_motion() function of the reference C model.</p>	31:24	Frame Level Global Motion Invalid Flags Format: U8
31:24	Frame Level Global Motion Invalid Flags Format: U8		
23	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">23</td> <td> Reserved (for future expansion of Global Motion Type[7]) Format: MBZ </td> </tr> </table>	23	Reserved (for future expansion of Global Motion Type[7]) Format: MBZ
23	Reserved (for future expansion of Global Motion Type[7]) Format: MBZ		
22:21	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">22:21</td> <td> Global Motion Type[7] Format: U2 </td> </tr> </table> <p>It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY (requires no additional warp projection parameter) Set to 1, specifies TRANSLATION (requires 2 additional warp projection parameters) Set to 2, specifies ROTZOOM (requires 4 additional warp projection parameters) Set to 3, specifies AFFINE (requires 6 additional warp projection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.</p>	22:21	Global Motion Type[7] Format: U2
22:21	Global Motion Type[7] Format: U2		
20	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">20</td> <td> Reserved (for future expansion of Global Motion Type[6]) Format: MBZ </td> </tr> </table>	20	Reserved (for future expansion of Global Motion Type[6]) Format: MBZ
20	Reserved (for future expansion of Global Motion Type[6]) Format: MBZ		
19:18	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">19:18</td> <td> Global Motion Type[6] Format: U2 </td> </tr> </table> <p>It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY (requires no additional projection parameter) Set to 1, specifies TRANSLATION (requires 2 additional projection parameters) Set to 2, specifies ROTZOOM (requires 4 additional projection parameters) Set to 3, specifies AFFINE (requires 6 additional projection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.</p>	19:18	Global Motion Type[6] Format: U2
19:18	Global Motion Type[6] Format: U2		
17	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">17</td> <td> Reserved (for future expansion of Global Motion Type[5]) </td> </tr> </table>	17	Reserved (for future expansion of Global Motion Type[5])
17	Reserved (for future expansion of Global Motion Type[5])		

AVP_PIC_STATE

	Format:	MBZ
16:15	Global Motion Type[5]	
	Format:	U2
	<p>It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame.</p> <p>Set to 0, specifies IDENTITY (requires no additional projection parameter)</p> <p>Set to 1, specifies TRANSLATION (requires 2 additional projection parameters)</p> <p>Set to 2, specifies ROTZOOM (requires 4 additional projection parameters)</p> <p>Set to 3, specifies AFFINE (requires 6 additional projection parameters)</p> <p>It is the frame level derived parameter <code>gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7]</code>, <code>gmtype[0]</code> is not being used.</p>	
14	Reserved (for future expansion of Global Motion Type[4])	
	Format:	MBZ
13:12	Global Motion Type[4]	
	Format:	U2
	<p>It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame.</p> <p>Set to 0, specifies IDENTITY (requires no additional projection parameter)</p> <p>Set to 1, specifies TRANSLATION (requires 2 additional projection parameters)</p> <p>Set to 2, specifies ROTZOOM (requires 4 additional projection parameters)</p> <p>Set to 3, specifies AFFINE (requires 6 additional projection parameters)</p> <p>It is the frame level derived parameter <code>gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7]</code>, <code>gmtype[0]</code> is not being used.</p>	
11	Reserved (for future expansion of Global Motion Type[3])	
	Format:	MBZ
10:9	Global Motion Type[3]	
	Format:	U2
	<p>It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame.</p> <p>Set to 0, specifies IDENTITY (requires no additional projection parameter)</p> <p>Set to 1, specifies TRANSLATION (requires 2 additional projection parameters)</p> <p>Set to 2, specifies ROTZOOM (requires 4 additional projection parameters)</p> <p>Set to 3, specifies AFFINE (requires 6 additional projection parameters)</p> <p>It is the frame level derived parameter <code>gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7]</code>, <code>gmtype[0]</code> is not being used.</p>	
8	Reserved (for future expansion of Global Motion Type[2])	
	Format:	MBZ
7:6	Global Motion Type[2]	
	Format:	U2

AVP_PIC_STATE			
	<p>It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame.</p> <p>Set to 0, specifies IDENTITY (requires no additional projection parameter)</p> <p>Set to 1, specifies TRANSLATION (requires 2 additional projection parameters)</p> <p>Set to 2, specifies ROTZOOM (requires 4 additional projection parameters)</p> <p>Set to 3, specifies AFFINE (requires 6 additional projection parameters)</p> <p>It is the frame level derived parameter $gmtype[LAST_FRAME \dots ALTREF_FRAME = 1..7]$, $gmtype[0]$ is not being used.</p>		
5	<p>Reserved (for future expansion of Global Motion Type[1])</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
4:3	<p>Global Motion Type[1]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame.</p> <p>Set to 0, specifies IDENTITY (requires no additional projection parameter)</p> <p>Set to 1, specifies TRANSLATION (requires 2 additional projection parameters)</p> <p>Set to 2, specifies ROTZOOM (requires 4 additional projection parameters)</p> <p>Set to 3, specifies AFFINE (requires 6 additional projection parameters)</p> <p>It is the frame level derived parameter $gmtype[LAST_FRAME \dots ALTREF_FRAME = 1..7]$, $gmtype[0]$ is not being used.</p>	Format:	U2
Format:	U2		
2:0	<p>Reserved (for future expansion of Global Motion Type[0])</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
9..29	<p>671:0 Warp Parameters Array [Reference=1 to 7][ProjectionCoeff=0 to 5]</p> <p>It specifies the Warp Parameter set for each of the 7 reference frames [LAST_FRAME ... ALTREF_FRAME]</p> <p>Each Warp Parameter set is contains of 6 warp projection coefficient [$projection_coeff = 0$ to 5]</p> <p>Each projection coefficient is a 16-bit signed integer (2's component).</p> <p>Total 7 references * 6 proj coeff / 2 = 21 Dwords.</p> <p>Different Global Motion Type is reading different number of projection coefficients from the bitstream. All projection coefficients are coded with $signed_subexp_with_ref()$. The number of bits read for each projection coefficients from the bitstream, depends on the Global Motion Type too. After decoded the $signed_subexp_with_ref()$, the projection coefficients are further upshifted and round to the final precision.</p> <p>Allowed range for each coeff is [-4096, 4096] for Warp Motion. For translation-only motion type, the max range for each coeff is [-512 to +512]. All the upper bits are assumed to have sign extended.</p> <p>For IDENTITY motion type, all the coefficients should set to 0.</p> <p>Dword 9 15:0 - Warp Parameters Array[1][0] Dword 9 31:16 -Warp Parameters Array[1][1] Dword 10 15:0 -Warp Parameters Array[1][2] Dword 10 31:16-Warp Parameters Array[1][3] ...</p>		

AVP_PIC_STATE				
		... Dword 29 15:0 -Warp Parameters Array[7][4] Dword 29 31:16-Warp Parameters Array[7][5]		
30	31:0	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
31	31:16	Intra Frame Height In Pixel Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U16</td> </tr> </table> <p>Specifies the height of the INTRA picture(Reference Picture0) in pixels. The INTRA picture height in units of luma samples equals (INTRA_FRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16
	Format:	U16		
15:0	Intra Frame Width In Pixel Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U16</td> </tr> </table> <p>Specifies the width of the INTRA picture(Reference Picture0) in pixels. The INTRA picture width in units of luma samples equals (INTRA_FRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16	
Format:	U16			
32	31:16	Last Frame Height In Pixel Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U16</td> </tr> </table> <p>Specifies the height of the LAST picture(Reference Picture1) in pixels. The LAST picture height in units of luma samples equals (LASTFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16
	Format:	U16		
15:0	Last Frame Width In Pixel Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U16</td> </tr> </table> <p>Specifies the width of the LAST picture(Reference Picture1) in pixels. The LAST picture width in units of luma samples equals (LASTFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16	
Format:	U16			
33	31:16	Last2 Frame Height In Pixel Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U16</td> </tr> </table> <p>Specifies the height of the LAST2(Reference Picture 2) picture in pixel. The LAST2 picture height in units of luma samples equals (LAST2FRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size.</p>	Format:	U16
Format:	U16			

AVP_PIC_STATE				
		<p>Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>		
	15:0	<p>Last2 Frame Width In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Specifies the width of the LAST2 picture(Reference Picture 2) in pixels. The LAST2 picture width in units of luma samples equals (LAST2FRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>	Format:	U16
Format:	U16			
34	31:16	<p>Last3 Frame Height In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Specifies the height of the LAST3 picture(Reference Picture 3) in pixels. The LAST3 picture height in units of luma samples equals (LAST3FRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>	Format:	U16
	Format:	U16		
15:0	<p>Last3 Frame Width In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Specifies the width of the LAST3 picture(Reference Picture 3) in pixels. The LAST3 picture width in units of luma samples equals (LAST3FRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>	Format:	U16	
Format:	U16			
35	31:16	<p>Golden Frame Height In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Specifies the height of the GOLDEN picture(Reference Picture 4) in pixels. The GOLDEN picture height in units of luma samples equals (GOLDENFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16
	Format:	U16		
15:0	<p>Golden Frame Width In Pixel Minus 1</p>			

AVP_PIC_STATE				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U16</td> </tr> </table> <p>Specifies the width of the GOLDEN picture(Reference Picture 4) in pixels. The GOLDEN picture width in units of luma samples equals (GOLDENFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16
Format:	U16			
36	31:16	<p>BWDREF Frame Height In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U16</td> </tr> </table> <p>Specifies the height of the BWDREF picture(Reference Picture 5) in pixels. The BWDREF picture height in units of luma samples equals (BWDREFFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16
	Format:	U16		
15:0	<p>BWDREF Frame Width In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U16</td> </tr> </table> <p>Specifies the width of the BWDREFpicture(Reference Picture 5) in pixels.The BWDREFpicture width in units of luma samples equals (BWDREFFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].</p>	Format:	U16	
Format:	U16			
37	31:16	<p>ALTREF2 Frame Height In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U16</td> </tr> </table> <p>Specifies the height of the ALTREF2 picture(Reference Picture 6) in pixels. The ALTREF picture height in units of luma samples equals (ALTREFFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>	Format:	U16
	Format:	U16		
15:0	<p>ALTREF2 Frame Width In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U16</td> </tr> </table> <p>Specifies the width of the ALTREF2 picture(Reference Picture 6) in pixels. The ALTREF picture width in units of luma samples equals (ALTREFFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfieldFrame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>	Format:	U16	
Format:	U16			

AVP_PIC_STATE				
38	31:16	<p>ALTREF Frame Height In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Specifies the height of the ALTREFpicture(Reference Picture 7) in pixels. The ALTREF2 picture height in units of luma samples equals $(ALTREF2FRAME_HeightInPixelMinus1 + 1)$ AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>	Format:	U16
	Format:	U16		
15:0	<p>ALTREF Frame Width In Pixel Minus 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Specifies the width of the ALTREFpicture(Reference Picture 7) in pixels. The ALTREF2 picture width in units of luma samples equals $(ALTREF2FRAME_WidthInPixelMinus1 + 1)$ AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfieldFrame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.</p>	Format:	U16	
Format:	U16			
39	31:16	<p>Horizontal Scale Factor for INTRA</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the INTRA reference frame (Reference Picture0). Set to $(INTRA_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
	Format:	U2.14		
15:0	<p>Vertical Scale Factor for INTRA</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the INTRA reference frame (Reference Picture 0). Set to $(INTRA_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14	
Format:	U2.14			
40	31:16	<p>Horizontal Scale Factor for LAST</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the LAST reference frame (Reference Picture1). Set to $(LAST_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
	Format:	U2.14		
15:0	<p>Vertical Scale Factor for LAST</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the LAST reference frame (Reference Picture 1). Set to $(LAST_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14	
Format:	U2.14			

AVP_PIC_STATE				
41	31:16	<p>Horizontal Scale Factor for LAST2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the LAST2 reference frame (Reference Picture 2). Set to $(LAST2_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$. Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
	Format:	U2.14		
15:0	<p>Vertical Scale Factor for LAST2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the LAST2 reference frame (Reference Picture 2). Set to $(LAST2_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$. Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14	
Format:	U2.14			
42	31:16	<p>Horizontal Scale Factor for LAST3</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the LAST3 reference frame (Reference Picture 3). Set to $(LAST3_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$. Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
	Format:	U2.14		
15:0	<p>Vertical Scale Factor for LAST3</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the LAST3 reference frame (Reference Picture 3). Set to $(LAST3_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$. Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14	
Format:	U2.14			
43	31:16	<p>Horizontal Scale Factor for GOLDEN</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the GOLDEN reference frame (Reference Picture 4). Set to $(GOLDEN_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$. Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
	Format:	U2.14		
15:0	<p>Vertical Scale Factor for GOLDEN</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the GOLDEN reference frame (Reference Picture 4). Set to $(GOLDEN_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$. Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14	
Format:	U2.14			
44	31:16	<p>Horizontal Scale Factor for BWDREF_FRAME</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the BWDREF_FRAME reference frame (Reference Picture 5). Set to $(BWDREF_FRAME_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$.</p>	Format:	U2.14
Format:	U2.14			

AVP_PIC_STATE				
		Scaling Factor can be [1/16 to 2].		
	15:0	<p>Vertical Scale Factor for BWDREF_FRAME</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the BWDREF_FRAME reference frame (Reference Picture 5). Set to $(BWDREF_FRAME_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
Format:	U2.14			
45	31:16	<p>Horizontal Scale Factor for ALTREF2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the ALTREF2 reference frame (Reference Picture 6). Set to $(ALTREF_FRAME_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
Format:	U2.14			
	15:0	<p>Vertical Scale Factor for ALTREF2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the ALTREF2 reference frame (Reference Picture 6). Set to $(ALTREF_FRAME_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
Format:	U2.14			
46	31:16	<p>Horizontal Scale Factor for ALTREF</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the ALTREF reference frame (Reference Picture 7). Set to $(ALTREF2_FRAME_Width * 2^{14} + (CurrentWidth / 2)) / CurrentWidth$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
Format:	U2.14			
	15:0	<p>Vertical Scale Factor for ALTREF</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the ALTREF reference frame (Reference Picture 7). Set to $(ALTREF2_FRAME_Height * 2^{14} + (CurrentHeight / 2)) / CurrentHeight$ Scaling Factor can be [1/16 to 2].</p>	Format:	U2.14
Format:	U2.14			
47	31:24	<p>Reference Frame Order Hint [3] for Last3 Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U8</td> </tr> </table> <p>It specifies the expected output order hint for each reference buffer. This is the LAST3 Reference Frame Order Hint (ReferenceFrame 3). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.</p>	Format:	U8
Format:	U8			

AVP_PIC_STATE			
	<p>23:16 Reference Frame Order Hint [2] for Last2 Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It specifies the expected output order hint for each reference buffer. This is the LAST2 Reference Frame Order Hint (ReferenceFrame 2). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.</p>	Format:	U8
	Format:	U8	
	<p>15:8 Reference Frame Order Hint [1] for Last Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It specifies the expected output order hint for each reference buffer. This is the LAST Reference Frame Order Hint (ReferenceFrame 1). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.</p>	Format:	U8
Format:	U8		
<p>7:0 Reference Frame Order Hint [0] for Intra Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It specifies the expected output order hint for each reference buffer. This is the INTRA Reference Frame Order Hint (ReferenceFrame 0). It is a derived frame-level parameter, which can be equal to 1) the frame-level syntax element, ref_order_hint[i=0 to 7], which is only present in the bitstream if the current frame type (frame-level syntax) is NOT a KEY Frame, AND frame level syntax element: error-resilient mode is set to 1 AND sequence level syntax element: enable_order_hint is set to 1. 2) OR, the saved order hint, when the reference frame was the current frame being decoded. Note : The values in the ref_order_hint array can be used to implement to gracefully handle cases when some frames have been lost. It is done at the SW level, not inside AVP HW pipeline.</p>	Format:	U8	
Format:	U8		
48	<p>31:24 Reference Frame Order Hint [7] for ALTREF Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It specifies the expected output order hint for each reference buffer. This is the ALTREF Reference Frame Order Hint (ReferenceFrame 7). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.</p>	Format:	U8
	Format:	U8	
	<p>23:16 Reference Frame Order Hint [6] for ALTREF2 Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It specifies the expected output order hint for each reference buffer. This is the ALTREF2 Reference Frame Order Hint (ReferenceFrame 6). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.</p>	Format:	U8
Format:	U8		
<p>15:8 Reference Frame Order Hint [5] for BWDREF Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table>	Format:	U8	
Format:	U8		

AVP_PIC_STATE				
		<p>It specifies the expected output order hint for each reference buffer. This is the BWDREF Reference Frame Order Hint (ReferenceFrame 5). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.</p>		
	7:0	<p>Reference Frame Order Hint [4] for Golden Frame</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It specifies the expected output order hint for each reference buffer. This is the GOLDEN Reference Frame Order Hint (ReferenceFrame 4). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.</p>	Format:	U8
Format:	U8			
49	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
50	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			

AVP_PIPE_BUF_ADDR_STATE

AVP_PIPE_BUF_ADDR_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The AVP Pipeline is selected with the Media Instruction Opcode "8h" for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>This state command provides the physical memory base addresses for all row store buffers, column store buffers (for tile column and in scalability mode), reconstructed output and reference frame buffers, and auxiliary data buffers (MV, segment map, etc.) that are required by the AV1 decoding and encoding process. This is a frame level state command and is shared by both encoding and decoding processes. AVP is a tile based pipeline and is a stateless pipeline, hence all sequence level, frame level, and segment level state commands must be resent to process each tile. Memory compression may be applicable to some of these buffers for BW saving. Note : there is no buffer to store the 16 QM table sets, they are implemented directly inside the HW pipeline.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = AVP = 3h	
22:16	Media Instruction Command		
	Default Value:	2h AVP_PIPE_BUF_ADDR_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	C0h		
1..16	511:0	Reference Frame Buffer Base Address (RefAddr[0-7])	
		Format:	SplitBaseAddress64ByteAligned[8]

AVP_PIPE_BUF_ADDR_STATE

This array specifies the physical memory base addresses of the 8 visible reference frame buffers using for inter-prediction. They all contain previously reconstructed/decoded frames. However, this Reference Frame Buffer array is a remapped version (see below the mapping equations) of the DPB buffer array specified in the AV1 Spec and Reference C Model. Intel implementation has converted the 2 level of reference frame indexing (using RefFrame[0..6] and ref_frame_idx[0..6]) into a single level indexing (using the enum INTRA_FRAME, LAST_FRAME, ..., ALTREF_FRAME, in this fixed order). In the process, the ref_frame_idx[0..6] array is eliminated by intel.

Application and Driver will continue to receive or parse the bitstream header based on the 2-level indexing AV1 DPB buffer array definition, which can contain more than 8 frame buffers, but only at most 8 of them are visible to the AVP HW pipeline and is in the form of single level indexing intel-remapped-DPB array.

Typically, these reference frame buffers are read-only, for the purpose of Motion Comp. Memory compression is applied in accessing these buffers.

At most only 7 out of the 8 visible reference frames can be used for Motion Comp. in decoding inter-coded blocks of the current frame. The subset of reference frames being used in decoding the current frame is setup by the Application and Driver. But it is recommended to set all Reference Frame Buffer Base Address to valid and known addresses for error handling.

AV1 Reference Frames are defined in the following order:

DW 0-1 RefAddr[0] - INTRA Frame (Reference Frame 0)

DW 2-3 RefAddr[1] - LAST Frame (Reference Frame 1) In AV1 Spec, it is mapped into DPB [ref_frame_idx[LAST_FRAME-LAST_FRAME]]

DW 4-5 RefAddr[2] - LAST2 Frame (Reference Frame 2) In AV1 Spec, it is mapped into DPB [ref_frame_idx[LAST2_FRAME-LAST_FRAME]]

DW 6-7 RefAddr[3] - LAST3 Frame (Reference Frame 3) In AV1 Spec, it is mapped into DPB [ref_frame_idx[LAST3_FRAME-LAST_FRAME]]

DW 8-9 RefAddr[4] - GOLDEN Frame (Reference Frame 4) In AV1 Spec, it is mapped into DPB [ref_frame_idx[GOLDEN_FRAME-LAST_FRAME]]

DW 10-11 RefAddr[5] - BWDREF Frame (Reference Frame 5) In AV1 Spec, it is mapped into DPB [ref_frame_idx[BWDREF_FRAME-LAST_FRAME]]

DW 12-13 RefAddr[6] - ALTREF2 Frame (Reference Frame 6) In AV1 Spec, it is mapped into DPB [ref_frame_idx[ALTREF2_FRAME-LAST_FRAME]]

DW 14-15 RefAddr[7] - ALTREF Frame (Reference Frame 7) In AV1 Spec, it is mapped into DPB [ref_frame_idx[ALTREF_FRAME-LAST_FRAME]]

Note : for 48 bit address, a pair of Dwords (i.e. 2) are needed to store each base address.

Note : This reference frame naming convention is a legacy specification in the reference C model. Although the very first reference frame is labeled as INTRA, it is not reserved only for a previously decoded KEY frame or a decoded INTRA-ONLY NON-KEY frame. Any newly decoded frame can be stored in any one of these 8 reference frame buffers for future reference - it is up to the Application and Driver that handle the DPB management according to the bitstream and normative rules specified in AV1.

Note : When IntraBC coding mode is used in the KEY Frame or in the INTRA-ONLY NON-KEY Frame, the intraBC coded block is decoded as a regular inter-coded block, and one of these 8 reference frame buffers will be directly used in a READ/WRITE fashion.

Note : the format of the pixels (Y, U and V components) stored inside these reference frame buffers is defined in the AV1_Surface_State Command. For monochrome video, all reference frame buffers can only have Luma Y component.

AVP_PIPE_BUF_ADDR_STATE				
		Note :All reference frame buffers' surface addresses must be 4K byte aligned. There is a max. of 8 Reference Picture Buffer Addresses, and all share the same third address DW in specifying 48-bit address.		
17	31:0	<p>Reference Frame Buffer Base Address Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MemoryAddressAttributes</td> </tr> </table> <p>All reference frame buffers' surface addresses must be 4K byte aligned. There is a max. of 8 Reference Picture Buffer Addresses, and all share this same third address DW in specifying 48-bit address.</p>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
18..19	63:0	<p>Decoded Output Frame Buffer Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>It specifies the physical memory base addresses of the frame buffer that stores the final pixel output of the AVP pipeline, just before the Film Grain unit(could be sent for display). Typically, it is a WRITE-only surface with pixel format specified in the AV1_SURFACE_STATE. It can be 4:0:0, 4:2:0, 4:2:2 or 4:4:4.</p> <p>Memory compression is applied to this surface.</p> <p>This buffer holds the result of reconstructing the current frame, including all the Post In-Loop Filtering Processes (Deblocker, CDEF, Super-Resolution, and/or Loop Restoration) that are enabled. There can be a few actions to follow:</p> <ol style="list-style-type: none"> 1) Application and Driver will then place this newly decoded frame into the Display Buffer (DPB), as one of the 8 reference frames, only if it will be used for the decoding of a later frame(s) 2) At the same time, if this newly decoded frame is to be displayed immediately, it will also be copied and sent to other system for further processing or directly to the Display Controller. 3) If Film Grain Injection is enabled in the sequence header, this newly decoded frame is further processed by the Out of Loop Film Grain Synthesis unit in a block-based pipeline fashion (continued from the block-based pixel reconstruction pipeline of the AV1 decoder). <p>Note: when intraBC is active, it can also be a READ surface for the Motion Comp operation.</p> <p>Note : there is only one write location along the in-loop decoding pipeline, and it is at the output of the Loop Restoration Filter. If any or all of the Post In-Loop Filters (Deblocker, CDEF, Super-Resolution, Loop Restoration)are disabled, the reconstructed pixels still need to pipethrough those disabled filters without change (i.e. simply bypass), until they reach the output of the Loop Restoration Filter.</p> <p>Note : if it is decided to implement some of the Post In-Loop Filters in software (as a separate pass - not being implemented yet), this buffer will hold the final pixel output frame of the HW pipeline.</p> <p>Note : this decoded output buffer if going to be used as a reference frame, then it is added into the Display Buffer (DPB) at the location specified by the frame level syntax element refresh_frame_flags. If the current frame is a KEY Frame or a SWITCH Frame, refresh_frame_flags is set to all 1's, so the entire DPB is initialized (KEY Frame)/re-initialized (SWITCH Frame) to the current frame.</p>	Format:	SplitBaseAddress4KByteAligned
Format:	SplitBaseAddress4KByteAligned			
20	31:0	<p>Decoded Output Frame Buffer Address Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			

AVP_PIPE_BUF_ADDR_STATE		
21..23	95:0	Reserved Format: MBZ
24..25	63:0	IntraBC Decoded Output Frame Buffer Address Format: SplitBaseAddress4KByteAligned When IntraBC (frame level) is ON, the current decoded and reconstructed partial pixel frame is needed for motion compensation. The normal Decoder Output Frame Buffer is typically written out with memory compression ON, as such it is not suitable to be read back for performing motion compensation within the decoding of the same frame. A separate decoded output frame buffer is needed, whose surface buffer does not have memory compression turned ON. That is, when IntraBC is ON (frame level), AVP will write out two decoded output frame with identical content.
26	31:0	IntraBC Decoded Output Frame Buffer Address Attributes Format: MemoryAddressAttributes
27..28	63:0	CDF Tables Initialization Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the CDF Tables Initialization Buffer. This Buffer is read-only. It is programmable with the initial CDF table set for the entire frame and for all its tiles at the very beginning before any decoding process. The content of this buffer must be the same for all tiles for the frame
29	31:0	CDF Tables Initialization Buffer Address Attributes Format: MemoryAddressAttributes
30..31	63:0	CDF Tables Backward Adaptation Buffer Address Format: SplitBaseAddress64ByteAligned Baseaddress of the CDF Tables Backward Adaptation Buffer. This Buffer stores the updated frame context of the largest tile in the current decoded frame. This is a WRITE-Only buffer.
32	31:0	CDF Tables Backward Adaptation Buffer Address Attributes Format: MemoryAddressAttributes
33..34	63:0	AV1 Segment ID Read Buffer Address Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for AV1 SegmentID buffer. This should contain the writeout SegmentID from previous frame and will be used to predict SegmentID for the current frame. Hardware will write out SegmentID of the current frame in the same address for the next frame. It is used for temporal prediction of segment ID in the current frame. The segment map has a granularity of 4x4 blocks.
35	31:0	AV1 Segment ID Read Buffer Address Attributes Format: MemoryAddressAttributes

AVP_PIPE_BUF_ADDR_STATE				
36..37	63:0	<p>AV1 Segment ID Write Buffer Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned buffer address for AV1 SegmentID buffer. This should contain the writeout SegmentID of the current frame and will be used to predict SegmentID for later frame. This segment map buffer is attached to the current decoded frame as its auxiliary data, and are both stored together in the Display Buffer (DPB), if the current frame is to be used as a reference frame to decode later frame(s). The segment map has a granularity of 4x4 blocks.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
38	31:0	<p>AV1 Segment ID Write Buffer Address Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
39..54	511:0	<p>Collocated Motion Vector Temporal Buffer Base Address (TmvAddr[0-7])</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>SplitBaseAddress64ByteAligned[8]</td> </tr> </table> <p>Base address for the Collocated Motion Vector Temporal buffer. The 8 Temporal Buffers are defined in the following order: DW 38-39 TmvAddr[0] - INTRA Frame (Reference Frame 0) DW 40..41 TmvAddr[1] - LAST Frame (Reference Frame 1) DW 42..43 TmvAddr[2] - LAST2 Frame (Reference Frame 2) DW 44..45 TmvAddr[3] - LAST3 Frame (Reference Frame 3) DW 46..47 TmvAddr[4] - GOLDEN Frame (Reference Frame 4) DW 48..49 TmvAddr[5] - BWDREF Frame (Reference Frame 5) DW 50..51 TmvAddr[6] - ALTREF2 Frame (Reference Frame 6) DW 52..53 TmvAddr[7] - ALTREF Frame (Reference Frame 7) Note : There is a max. of 8 Collocated MV TemporalBuffer Addresses, and all share the same third address DW in specifying 48-bit address</p>	Format:	SplitBaseAddress64ByteAligned[8]
Format:	SplitBaseAddress64ByteAligned[8]			
55	31:0	<p>Collocated Motion Vector Temporal Buffer Base Address Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MemoryAddressAttributes</td> </tr> </table> <p>Note : There is a max. of 8 Collocated MV TemporalBuffer Addresses, and all share this same third address DW in specifying 48-bit address.</p>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
56..57	63:0	<p>Current Frame Motion Vector Write Buffer Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address for the Current Motion Vector Temporal buffer.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
58	31:0	<p>Current Frame Motion Vector Write Buffer Address Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
59..61	95:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
62..63	63:0	<p>Bitstream Decoder/Encoder Line Rowstore Read/Write Buffer Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned buffer address for Bitstream Decode Line Rowstore</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			

AVP_PIPE_BUF_ADDR_STATE		
64	31:0	Bitstream Decoder/Encoder Line Rowstore Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
65..66	63:0	Bitstream Decoder/Encoder Tile Line Rowstore Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for Bitstream Decode Tile Line Buffer
67	31:0	Bitstream Decoder/Encoder Tile Line Rowstore Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
68..69	63:0	Intra Prediction Line Rowstore Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for Intra Prediction Line Rowstore
70	31:0	Intra Prediction Line Rowstore Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
71..72	63:0	Intra Prediction Tile Line Rowstore Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for Intra Prediction Tile Line Rowstore
73	31:0	Intra Prediction Tile Line Rowstore Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
74..75	63:0	Spatial Motion Vector Line Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Spatial Motion Vector Line buffer.
76	31:0	Spatial Motion Vector Line Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
77..78	63:0	Spatial Motion Vector Coding Tile Line Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Spatial Motion Vector Tile Line buffer.
79	31:0	Spatial Motion Vector Tile Line Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
80..81	63:0	Loop Restoration Meta Tile Column Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base Address for Loop Restoration Meta Tile Column Read/Write Buffer
82	31:0	Loop Restoration Meta Tile Column Read/Write Buffer Address Attributes

AVP_PIPE_BUF_ADDR_STATE		
		Format: MemoryAddressAttributes
83..84	63:0	Loop Restoration Filter Tile Read/Write Line Y Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Line Y Buffer
85	31:0	Loop Restoration Filter Tile Read/Write Line Y Buffer Address Attributes Format: MemoryAddressAttributes
86..87	63:0	Loop Restoration Filter Tile Read/Write Line U Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Line U Buffer
88	31:0	Loop Restoration Filter Tile Read/Write Line U Buffer Address Attributes Format: MemoryAddressAttributes
89..90	63:0	Loop Restoration Filter Tile Read/Write Line V Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Line V Buffer
91	31:0	BitField: Loop Restoration Filter Tile Read/Write Line V Buffer Address Attributes Format: MemoryAddressAttributes
92..93	63:0	Deblocker Filter Line Read/Write Y Buffer Address Format: SplitBaseAddress64ByteAligned Base address of the filter line buffer (read/write) used by the Deblocking Filter.
94	31:0	Deblocker Filter Line Read/Write Y Buffer Address Attributes Format: MemoryAddressAttributes
95..96	63:0	Deblocker Filter Line Read/Write U Buffer Address Format: SplitBaseAddress64ByteAligned Base address of the filter line buffer (read/write) used by the Deblocking Filter.
97	31:0	Deblocker Filter Line Read/Write U Buffer Address Attributes Format: MemoryAddressAttributes
98..99	63:0	Deblocker Filter Line Read/Write V Buffer Address Format: SplitBaseAddress64ByteAligned Base address of the filter line buffer (read/write) used by the Deblocking Filter.
100	31:0	Deblocker Filter Line Read/Write V Buffer Address Attributes Format: MemoryAddressAttributes
101..102	63:0	Deblocker Filter Tile Line Read/Write Y Buffer Address

AVP_PIPE_BUF_ADDR_STATE				
		<table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile line buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
103	31:0	Deblocker Filter Tile Line Read/Write Y Buffer Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
104..105	63:0	Deblocker Filter Tile Line Read/Write V Buffer Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile line buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
106	31:0	Deblocker Filter Tile Line Read/Write V Buffer Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
107..108	63:0	Deblocker Filter Tile Line Read/Write U Buffer Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile line buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
109	31:0	Deblocker Filter Tile Line Read/Write U Buffer Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
110..111	63:0	Deblocker Filter Tile Column Read/Write Y Buffer Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile column buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
112	31:0	Deblocker Filter Tile Column Read/Write Y Buffer Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
113..114	63:0	Deblocker Filter Tile Column Read/Write U Buffer Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile column buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
115	31:0	Deblocker Filter Tile Column Read/Write U Buffer Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
116..117	63:0	Deblocker Filter Tile Column Read/Write V Buffer Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile column buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
118	31:0	Deblocker Filter Tile Column Read/Write V Buffer Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
119..120	63:0	CDEF Filter Line Read/Write Buffer Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address for the CDEF Filter Line buffer. It include YUV data</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			

AVP_PIPE_BUF_ADDR_STATE		
121	31:0	CDEF Filter Line Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
122..127	191:0	Reserved Format: MBZ
128..129	63:0	CDEF Filter Tile Line Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the CDEF Filter Tile Line buffer. It includes YUV data
130	31:0	CDEF Filter Tile Line Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
131..136	191:0	Reserved Format: MBZ
137..138	63:0	CDEF Filter Tile Column Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the CDEF Filter Tile Column buffer. It includes YUV data
139	31:0	CDEF Filter Tile Column Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
140..141	63:0	CDEF Filter Meta Tile Line Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the CDEF Filter Meta Tile Line buffer.
142	31:0	CDEF Filter Meta Tile Line Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
143..144	63:0	CDEF Filter Meta Tile Column Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the CDEF Filter Meta Tile Column buffer.
145	31:0	CDEF Filter Meta Tile Column Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
146..147	63:0	CDEF Filter Top-Left Corner Read/Write Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the CDEF Filter Tile Column buffer.
148	31:0	CDEF Filter Top-Left Corner Read/Write Buffer Address Attributes Format: MemoryAddressAttributes
149..150	63:0	Super-Res Tile Column Read/Write Y Buffer Address Format: SplitBaseAddress64ByteAligned

AVP_PIPE_BUF_ADDR_STATE		
		Base address for the Super-ResolutionTile Column buffer.
151	31:0	Super-Res Tile Column Read/Write Y Buffer Address Attributes Format: MemoryAddressAttributes
152..153	63:0	Super-Res Tile Column Read/Write U Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Super-Resolution Tile Column buffer.
154	31:0	Super-Res Tile Column Read/Write U Buffer Address Attributes Format: MemoryAddressAttributes
155..156	63:0	Super-Res Tile Column Read/Write V Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Super-Resolution Tile Column buffer.
157	31:0	Super-Res Tile Column Read/Write V Buffer Address Attributes Format: MemoryAddressAttributes
158..159	63:0	Loop Restoration Filter Tile Column Read/Write Y Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Column buffer.
160	31:0	Loop Restoration Filter Tile Column Read/Write Y Buffer Address Attributes Format: MemoryAddressAttributes
161..162	63:0	Loop Restoration Filter Tile Column Read/Write U Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Column buffer.
163	31:0	Loop Restoration Filter Tile Column Read/Write U Buffer Address Attributes Format: MemoryAddressAttributes
164..165	63:0	Loop Restoration Filter Tile Column Read/Write V Buffer Address Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Column buffer.
166	31:0	Loop Restoration Filter Tile Column Read/Write V Buffer Address Attributes Format: MemoryAddressAttributes
167..169	95:0	Reserved Format: MBZ
170..175	191:0	Reserved Format: MBZ

AVP_PIPE_BUF_ADDR_STATE						
176..177	63:0	<p>Decoded Frame Status/Error Buffer Base Address</p> <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Decoder Mode : Specifies the 64 byte aligned buffer address for writing a single status/error cache-line sized record into memory when the Pic Status/Error Report Enable is set in the AVP_PIPE_MODE_SELECT command. The pic status/error record is written by hardware after the picture is decoded. The content of this memory location is not encrypted and can later be read by the driver only.</p>	Format:	SplitBaseAddress64ByteAligned		
Format:	SplitBaseAddress64ByteAligned					
178	31:0	<p>Decoded Frame Status/Error Buffer Base Address Attributes</p> <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes		
Format:	MemoryAddressAttributes					
179..180	63:0	<p>Decoded Block Data Streamout Buffer Address</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Buffer address for outputting the per-block indirect data to memory when StreamOutEnable is set in the AVP_PIPE_MODE_SELECT command.</p> <p>For Decoder: this field is used for transcoding purpose.</p>	Exists If:	//Decoder Only	Format:	SplitBaseAddress64ByteAligned
Exists If:	//Decoder Only					
Format:	SplitBaseAddress64ByteAligned					
181	31:0	<p>Decoded Block Data Streamout Buffer Address Attributes</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Exists If:	//Decoder Only	Format:	MemoryAddressAttributes
Exists If:	//Decoder Only					
Format:	MemoryAddressAttributes					
182..184	95:0	Reserved				
185..187	95:0	Reserved				

AVP_PIPE_MODE_SELECT

AVP_PIPE_MODE_SELECT			
Source:	VideoCS		
Length Bias:	2		
<p>The AVP Pipeline is selected with the Media Instruction Opcode "8h" for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The workload for the AVP pipeline is tile based. Once the bit stream DMA is configured with the AVP_BSD_OBJECT command for a tile in a frame, and the tile's bitstream is presented to the AVP, the tile decoding will begin.</p> <p>AVP pipeline is stateless, i.e. there is no states saved between the decoding of each tile. Hence all sequence, frame and segment state commands have to be resent before the tile coding command and the BSD object command.</p> <p>The AVP_PIPE_MODE_SELECT command is responsible for general pipeline level configuration that would normally be set once for a single stream encode or decode and would not be modified on a frame workload basis.</p> <p>This is a frame level state command and is shared by both encoding and decoding processes.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = AVP = 3h	
22:16	Media Instruction Command		
	Default Value:	0h AVP_PIPE_MODE_SELECT	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	4h		

AVP_PIPE_MODE_SELECT

1	31:24	Reserved		
	23	Reserved		
	22:17	Reserved		
		Format:	MBZ	
	16:15	Pipe working Mode		
		This programs the working mode for AVP pipe.		
		Value	Name	Description
		00b	Legacy decoder/encoder mode (Single pipe)	This is for single pipe mode non-scalable mode. It is used by both decoder and encoder.
		01b	Reserved	
		11b	Decoder Scalable mode with MSAC in real tiles (Scalable Multi-pipe)	This is for multiple-pipe scalable mode decoder mode in real tiles. MSAC and reconstruction will run together. Each pipes will run in real tiles vertically.
14:13	Multi-Engine Mode			
	This indicates the current pipe is in single pipe mode or if in scalable mode is in left/right/middle pipe in multi-engine mode.			
	Value	Name	Description	
	00b	Single Engine Mode or MSAC FE only decode mode	This is for single engine mode (legacy) OR MSAC FE only decode mode During AV1Decoder Scalability Real Tile Mode, for the last phase, it is possible to have single tile column left. In this case, it should be programmed with pipe as a single engine mode (using this value). For example, for 9 tile column running on 4 pipes. The first two phases will use all 4 pipes and finish 8 tile column. The remaining one column will be processed as last third phase as single tile column.	
	01b	Pipe is the left engine in a Multi-engine mode	Current pipe is the most left engine while running in scalable multi-engine mode	
	10b	Pipe is the right engine in a Multi-engine mode	Current pipe is the most right engine while running in scalable multi-engine mode	
	11b	Pipe is one of the middle engine in a Multi-engine mode	Current pipe is in one of the middle engine while running in scalable multi-engine mode	
12	Reserved			
	Format:	MBZ		
11	Reserved			
	Format:	MBZ		

AVP_PIPE_MODE_SELECT										
10	Reserved	Format: MBZ								
	9:8	Reserved								
	9:8	Format: MBZ								
	7:5	Codec Standard Select								
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">2</td> <td>AV1</td> </tr> </tbody> </table>	Value	Name	2	AV1				
	Value	Name								
	2	AV1								
	4	Reserved	Format: MBZ							
	3	Pic Status/Error Report Enable	Format: Enable							
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> <td>Disable status/error reporting</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Enable</td> <td>Status/Error reporting is written out once per picture. The Pic Status/Error Report ID in DWord3 along with the status/error status bits are packed into one cache line and written to the Status/Error Buffer address in the AVP_PIPE_BUF_ADDR_STATE command. Must be zero for encoder mode.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Disable status/error reporting	1	Enable
Value	Name	Description								
0	Disable	Disable status/error reporting								
1	Enable	Status/Error reporting is written out once per picture. The Pic Status/Error Report ID in DWord3 along with the status/error status bits are packed into one cache line and written to the Status/Error Buffer address in the AVP_PIPE_BUF_ADDR_STATE command. Must be zero for encoder mode.								
2:1	Reserved	Format: MBZ								
0	Codec Select	Format: U1								
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Decode</td> </tr> </tbody> </table>	Value	Name	0	Decode					
Value	Name									
0	Decode									
2	31:0	Reserved								
	31:0	Format: MBZ								
3	31:0	Pic Status/Error Report ID								
	31:0	Format: U32								
	<p>The Pic Status/Error Report ID is a unique 32-bit unsigned integer assigned to each picture status/error output. Must be zero for encoder mode.</p>									
	Programming Notes									
	<p>Software must program different Status/Error Buffer addresses between pictures; otherwise the hardware might overwrite previously written data.</p>									
4	31:0	Reserved								
	31:0	Format: MBZ								
5	31:0	Reserved								
	31:0	Format: MBZ								

AVP_SEGMENT_STATE

AVP_SEGMENT_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>When segmentation is enabled, this Segment State command is issued once per segment. There can be maximum 8 segments specified to decode a given frame, so this Segment State Command can be issued maximum 8 times. It is assumed that there is no gap in segment IDs. So, when the AVP PIC States specified that the number of active</p> <p>When segmentation is disabled, driver still sends out this command once for segment id = 0. HW needs to check the segmentation enable flag from AVP_PIC_STATE Command as well to distinguish from the case when segmentation is enabled for segment id = 0.</p> <p>Each segment can have its own specification of enabling any one of the 8 features defined in AV1 and their corresponding feature data. When a feature is not enabled, its feature data is defaulted to 0. When segmentation is not enabled, all the features are disabled and their corresponding feature data are set to 0.</p> <p>Segment State Command also provides other segment related parameters.</p> <p>It is assumed that HW is keeping a copy of the complete AV1 QM Matrix Table for all color components inside its internal memory, and Driver only needs to send the qm_level as index into this Table.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = AVP = 3h	
	22:16	Media Instruction Command	
		Default Value:	32h AVP_SEGMENT_STATE
Format:		OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
2h			

AVP_SEGMENT_STATE				
1	31:3	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
2:0	<p>Segment ID</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U3</td> </tr> </table> <p>The Segment ID specifies which one of the 8 possible segments that the current Segment State Command is associated with. Segment ID is in the range of [0 ... 7]. Maximum, there can be 8 segments specified for decoding a given frame. Segment ID=0, even when segmentation is disabled.</p>	Format:	U3	
Format:	U3			
2	31:28	<p>Segment Chroma V QM Level</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U4</td> </tr> </table> <p>It specifies one of the 16 QM matrices to be used for the Chroma V component of the current segment. It is in the range of [0..15]. QM Level = 15, is a bypass, with no scaling. Default is 15. If the current segment is coded as lossless or if qm matrix is not enabled (frame level SE : using_qmatrix = = 0), then Segment QM Level is set to 15. Chroma V qmlevel = (segment lossless flag using_qmatrix == 0) ? 15: qm_v (frame level SE). If separate_uv_delta_q (sequence level SE) is set to 0, Segment Chroma V QM Level is set to the same value as of Segment Chroma UQM Level. If separate_uv_delta_q (sequence level SE) is set to 1, Segment Chroma V QM Level can be set independently to a different value as of Segment Chroma U QM Level. For a given frame, all lossy segments should have the same value for Vqm_level.</p>	Format:	U4
	Format:	U4		
	27:24	<p>Segment Chroma U QM Level</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U4</td> </tr> </table> <p>It specifies one of the 16 QM matrices to be used for the Chroma U component of the current segment. It is in the range of [0..15]. QM Level = 15, is a bypass, with no scaling. Default is 15. If the currentsegment is coded as lossless or if qm matrix is not enabled ((frame level SE : using_qmatrix = = 0), then Segment QM Level is set to 15. Chroma U qmlevel = (segment lossless flag using_qmatrix == 0) ? 15: qm_u (frame level SE). For a given frame, all lossy segments should have the same value for Uqm_level.</p>	Format:	U4
Format:	U4			
23:20	<p>Segment Luma Y QM Level</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U4</td> </tr> </table> <p>It specifies one of the 16 QM matrices to be used for the Luma Y component of the current segment. It is in the range of [0..15]. QM Level = 15, is a bypass, with no scaling. Default is 15. If the current segment is coded as lossless or if qm matrix is not enabled (frame level SE : using_qmatrix = = 0), then Segment QM Level is set to 15. Luma qmlevel = (segment lossless flag using_qmatrix == 0) ? 15: qm_y (frame level SE). For a given frame, all lossy segments should have the same value for Y qm_level.</p>	Format:	U4	
Format:	U4			

AVP_SEGMENT_STATE			
19	<p>Segment Lossless Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>0 : the current segment is not coded as lossless.Default is 0. 1 : the current segment is coded as lossless. Segment Lossless Flag = (clamp[(base_qindex + optional segment delta qindex), 0, 255] == 0) && (y_dc_delta_q == 0) && (u_dc_delta_q == 0) && (u_ac_delta_q == 0) && (v_dc_delta_q == 0) && (v_ac_delta_q == 0) It is computing usingsyntax elements all from the bitstream, read in the uncompressed header.</p>	Format:	U1
Format:	U1		
18	<p>Segment Block GlobalMV Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id = SEG_LVL_GLOBALMV = 7. Segment Block GlobalMV Flag = 0, specifies nothing. Segment Block GlobalMV Flag = 1, specifies the blocks of the current segment are all coded with Y inter prediction mode set to GLOBALMV. There is no feature data read from the bitstream for setting the Segment Block GlobalMV Flag. It is derived from it corresponding segment feature mask. Segment Block GlobalMV Flag =seg_feature_mask & (1 « SEG_LVL_GLOBALMV) If Segment Block GlobalMV Flag = 1, The Y inter prediction mode is set to ZeroMV (for using zero MV), and RefFrame[0] (??? elaborate) is set to LAST_FRAME, and RefFrame[1] is set to NONE (-1).</p>	Format:	U1
Format:	U1		
17	<p>Segment Block Skip Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id = SEG_LVL_SKIP = 6. Segment Block Skip Flag = 0, specifies there is no segment block skip. Segment Block Skip Flag = 1, specifies the blocks of the current segment are all skipped. These blocks are then having all coefficients and MV set to 0 (Y inter prediction mode is set to zeroMV mode), but these blocks can still have coding mode specified in the bitstream. There is no feature data read from the bitstream for setting the Segment Block Skip Flag. It is derived from it corresponding segment feature mask. Segment Block Skip Flag =seg_feature_mask & (1 « SEG_LVL_SKIP) If Segment Block Skip Flag = 1, no coefficient is present in the bitstream, and are all set to 0. The Y inter prediction mode is set to ZeroMV (for using zero MV), and RefFrame[0] is set to LAST_FRAME (??? elaborate) , and RefFrame[1] is set to NONE (-1). It is used to derived the skip_mode flag and skip flag.</p>	Format:	U1
Format:	U1		
16:8	<p>Segment Delta Qindex</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S8</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_Q = 0. It is a 9-bit signed 2's complement number, in the range of [-255, +255]. Default is 0. Value -256 is not allowed.</p>	Format:	S8
Format:	S8		
7:0	<p>Segment Feature Mask</p>		

AVP_SEGMENT_STATE							
	<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>It is an 8-bit mask specifies which of the 8 segment features are active in the current segment specifications. It is also known as feature enable mask or feature active mask. This mask can indicate 0 or up to 8 features are enabled. Any feature can be active or inactive independent of the other features.</p> <p>$\text{seg_feature_mask} \& (1 \ll \text{feature_id}) = 0$, the feature (identified by feature_id) is not enabled/active in the current segment.</p> <p>$\text{seg_feature_mask} \& (1 \ll \text{feature_id}) = 1$, the feature (identified by feature_id) is enabled/active in the current segment.</p> <p>Feature_ID (enum) Feature Name</p> <p>0 SEG_LVL_ALT_Q, // Use alternate Quantizer.</p> <p>1 SEG_LVL_ALT_LF_Y_V, // Use alternate loop filter value on y plane vertical.</p> <p>2 SEG_LVL_ALT_LF_Y_H, // Use alternate loop filter value on y plane horizontal.</p> <p>3 SEG_LVL_ALT_LF_U, // Use alternate loop filter value on u plane.</p> <p>4 SEG_LVL_ALT_LF_V, // Use alternate loop filter value on v plane.</p> <p>5 SEG_LVL_REF_FRAME, // Optional Segment reference frame.</p> <p>6 SEG_LVL_SKIP, // Optional Segment zeroMV (0,0) + skip mode.</p> <p>7 SEG_LVL_GLOBALMV. // Optional Segment zeroMV (0,0).</p> <p>Each segment has its own seg_feature_mask.</p> <p>When a segment is not being used or when segmentation is disabled, the corresponding seg_feature_mask has all 8-bits set to 0. Default all 8-bits are 0.</p>	Format:	U8				
Format:	U8						
3	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">31</td> <td style="width: 10%;">Reserved</td> <td style="width: 80%;"></td> </tr> <tr> <td></td> <td>Format:</td> <td>MBZ</td> </tr> </table>	31	Reserved			Format:	MBZ
31	Reserved						
	Format:	MBZ					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">30:28</td> <td style="width: 10%;">Segment Reference Frame</td> <td style="width: 80%;"></td> </tr> <tr> <td></td> <td>Format:</td> <td>U3</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id =SEG_LVL_REF_FRAME = 5. It is an alternate specification of one of the 8 possible reference frames for a Motion Compensation.</p> <p>=0 for INTRA_FRAME</p> <p>=1 for LAST_FRAME</p> <p>=2 for LAST2_FRAME</p> <p>=3 for LAST3_FRAME</p> <p>=4 for GOLDEN_FRAME</p> <p>=5 for BWDREF_FRAME</p> <p>=6 for ALTREF2_FRAME</p> <p>=7 for ALTREF_FRAME</p> <p>It is in the range of [0..7]. Default is 0.</p>	30:28	Segment Reference Frame			Format:	U3
30:28	Segment Reference Frame						
	Format:	U3					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 10%; text-align: center;">27:21</td> <td style="width: 10%;">Segment Delta Loop Filter Level Chroma V</td> <td style="width: 80%;"></td> </tr> <tr> <td></td> <td>Format:</td> <td>S6</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_V= 4. It is associated with filter_level_v read from the uncompressed header.</p> <p>It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.</p>	27:21	Segment Delta Loop Filter Level Chroma V			Format:	S6
27:21	Segment Delta Loop Filter Level Chroma V						
	Format:	S6					

AVP_SEGMENT_STATE			
20:14	<p>Segment Delta Loop Filter Level Chroma U</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>S6</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_U= 3. It is associated with filter_level_u read from the uncompressed header. It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.</p>	Format:	S6
Format:	S6		
13:7	<p>Segment Delta Loop Filter Level Luma Horizontal</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>S6</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_Y_H= 2. It is associated with filter_level[1] read from the uncompressed header. It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.</p>	Format:	S6
Format:	S6		
6:0	<p>Segment Delta Loop Filter Level Luma Vertical</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>S6</td> </tr> </table> <p>It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_Y_V = 1. It is associated with filter_level[0] read from the uncompressed header. It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.</p>	Format:	S6
Format:	S6		

AVP_SURFACE_STATE

AVP_SURFACE_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The AVP Pipeline is selected with the Media Instruction Opcode "8h" for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The AVP_SURFACE_STATE command is responsible for defining the frame buffer pitch and the offset of the chroma component.</p> <p>This is a picture level state command and is shared by both encoding and decoding processes.</p> <p>For Decoder, this command is issued once per surface type. There is one reconstructed surface, 8 reference pictures surfaces and one optional IntraBC Decoded Surface (only if IBC is ON).</p> <p>For Encoder, this command is issued once per surface type. There are 4 surface types :source down scaled, source original, reference and reconstructed picture. All reference frames are defined with the same surface command.</p> <p>Tile-Yf and Tile-Ys are not supported, but HW interface still need to keep these bits as reserved bits.</p> <p>Note : When NV12 and Tile Y are being used, full pitch and interleaved UV is always in use. U and V Xoffset must be set to 0; U and V Yoffset must be 8-pixel aligned. For 10-bit pixel, P010 surface definition is being used.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name
		Format:	OpCode
	Codec/Engine Name = AVP = 3h		
22:16	Media Instruction Command		
	Default Value:	1h SURFACE_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
		Value	
		Name	

AVP_SURFACE_STATE

AVP_SURFACE_STATE					
		1h			
1	31:28	Surface Id			
		Format: U4			
		Value	Name	Description	Programming Notes
		0h	Reconstructed Picture	This is for the reconstructed picture surface state	
		6h	AV1 INTRA FRAME	This is for AV1 Intra Frame (Reference Picture 0). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		7h	AV1 Last Frame	This is for AV1Last Frame (Reference Picture 1). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		8h	AV1 Last2 Frame	This is for AV1 Last2 Frame (Reference Picture 2). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		9h	AV1 Last3 Frame	This is for AV1 Last3 Frame (Reference Picture 3). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		Ah	AV1 Golden Frame	This is for AV1 Golden Frame (Reference Picture 4). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		Bh	AV1 Bwdref Frame	This is for AV1 Bwdref Frame (Reference Picture 5). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		Ch	AV1 Altref2 Frame	This is for AV1 Altref2 Frame (Reference Picture 6). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		Dh	AV1 Altref Frame	This is for AV1 Altref Frame (Reference Picture 7). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
		Eh	IntraBC Decoded Frame	This is for AV1 IntraBC Decoded Frame. It will be used both Read/Write at the same time. This surface requires multiple of 8 pixels on both width and height.	This surface must be programmed as uncompressed
	27:17	Reserved			
		Format: MBZ			
	16:0	Surface Pitch Minus1			
		Format: U17-1			
		This field specifies the surface pitch in (#Bytes - 1).			
		Programming Notes			

AVP_SURFACE_STATE									
	<p>For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case). The range in bytes is $[2^{Cu}-1, 131071] \rightarrow [(2^{Cu})B, 128KB] = [1 \text{ tile}, 128KB/(2^{Cu} \text{ tiles})]$ The field specifies the surface pitch in (#Bytes - 1)</p> <p>For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]</p>								
2	<p>31:27 Surface Format</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U5</td> </tr> </table> <p>Specifies the format of the surface.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 40%;">Value</th> <th style="width: 60%;">Name</th> </tr> </thead> <tbody> <tr> <td>4h</td> <td>PLANAR_420_8</td> </tr> <tr> <td>Dh</td> <td>P010</td> </tr> </tbody> </table>	Format:	U5	Value	Name	4h	PLANAR_420_8	Dh	P010
Format:	U5								
Value	Name								
4h	PLANAR_420_8								
Dh	P010								
26	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
25	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
24:15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
14:0	<p>Y Offset for U(Cb) in pixel</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U15</td> </tr> </table> <p>This field specifies the vertical offset in rows from the Surface Base Address to the start(origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr style="background-color: #e1eef6;"> <th>Programming Notes</th> </tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the Memory Address Attributes table. TileY (legacy 4k) - 8 pixel aligned TileYF (New 4k) - 64 pixel aligned TileYS (64k) - 256 pixel aligned </td> </tr> </tbody> </table>	Format:	U15	Programming Notes	<ul style="list-style-type: none"> For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the Memory Address Attributes table. TileY (legacy 4k) - 8 pixel aligned TileYF (New 4k) - 64 pixel aligned TileYS (64k) - 256 pixel aligned 				
Format:	U15								
Programming Notes									
<ul style="list-style-type: none"> For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the Memory Address Attributes table. TileY (legacy 4k) - 8 pixel aligned TileYF (New 4k) - 64 pixel aligned TileYS (64k) - 256 pixel aligned 									
3	<p>31:16 Y Offset for V(Cr)</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Row Offset in Pixels</p> <p>This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr style="background-color: #e1eef6;"> <th>Programming Notes</th> </tr> </thead> <tbody> <tr> <td> </td> </tr> </tbody> </table>	Format:	U16	Programming Notes					
Format:	U16								
Programming Notes									

AVP_SURFACE_STATE							
	<ul style="list-style-type: none"> • TileY (legacy 4k) - 8 pixel aligned • TileYF (New 4k) - 64 pixel aligned • TileYS (64k) - 256 pixel aligned 						
	15:0 Default Alpha Value						
4	31:21 Reserved						
	Format: MBZ						
	20:16 Reserved						
	Format: MBZ						
	15 Compression Type for Altref Frame This bit is for AV1 Altref Frame (Reference Picture 7). Valid only when Memory Compression for Altref Frame is enabled.						
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render compression Enabled
	Value	Name					
	0	Media compression Enabled [Default]					
1	Render compression Enabled						
14 Compression Type for Altref2 Frame This bit is for AV1 Altref2 Frame (Reference Picture 6). Valid only when Memory Compression for Altref2 Frame is enabled.							
<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled	
Value	Name						
0	Media compression Enabled [Default]						
1	Render Compression Enabled						
13 Compression Type for Bwdref Frame This bit is for AV1 Bwdref Frame (Reference Picture 5). Valid only when Memory Compression for Bwdref Frame is enabled.							
<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled	
Value	Name						
0	Media compression Enabled [Default]						
1	Render Compression Enabled						
12 Compression Type for Golden Frame This bit is for AV1 Golden Frame (Reference Picture 4). Valid only when Memory Compression for Golden Frame is enabled.							
<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled	
Value	Name						
0	Media compression Enabled [Default]						
1	Render Compression Enabled						
11 Compression Type for Last3 Frame This bit is for AV1 Last3 Frame (Reference Picture 3). Valid only when Memory Compression for Last3 Frame is enabled.							
<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled	
Value	Name						
0	Media compression Enabled [Default]						
1	Render Compression Enabled						

AVP_SURFACE_STATE

10	<p>Compression Type for Last2 Frame This bit is for AV1 Last2 Frame (Reference Picture 2). Valid only when Memory Compression for Last2 Frame is enabled.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled
Value	Name						
0	Media compression Enabled [Default]						
1	Render Compression Enabled						
9	<p>Compression Type for Last Frame This bit is for AV1 Last Frame (Reference Picture 1). Valid only when Memory Compression for Last Frame is enabled.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled
Value	Name						
0	Media compression Enabled [Default]						
1	Render Compression Enabled						
8	<p>Compression Type for Intra Frame This bit is for Intra Frame (Reference Picture 0). Valid only when Memory Compression for Intra Frame is enabled.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled
Value	Name						
0	Media compression Enabled [Default]						
1	Render Compression Enabled						
7	<p>Memory Compression Enable for AV1 Altref Frame This bit is for AV1Altref Frame (Reference Picture 7).</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable
Value	Name						
1	Memory Compression Enable						
0	Memory Compression Disable						
6	<p>Memory Compression Enable for AV1 Altref2 Frame This bit is for AV1Altref2 Frame (Reference Picture 6).</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable
Value	Name						
1	Memory Compression Enable						
0	Memory Compression Disable						
5	<p>Memory Compression Enable for AV1 Bwdref Frame This bit is for AV1 Bwdref Frame (Reference Picture 5).</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable
Value	Name						
1	Memory Compression Enable						
0	Memory Compression Disable						
4	<p>Memory Compression Enable for AV1 Golden Frame This bit is for AV1 Golden Frame (Reference Picture 4).</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable
Value	Name						
1	Memory Compression Enable						
0	Memory Compression Disable						

AVP_SURFACE_STATE								
	3	Memory Compression Enable for AV1 Last3 Frame This bit is for AV1 Last3 Frame (Reference Picture 3).						
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable
	Value	Name						
	1	Memory Compression Enable						
	0	Memory Compression Disable						
	2	Memory Compression Enable for AV1 Last2 Frame This bit is for AV1 Last2 Frame (Reference Picture 2).						
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable
	Value	Name						
	1	Memory Compression Enable						
	0	Memory Compression Disable						
	1	Memory Compression Enable for AV1 Last Frame This bit is for AV1 Last Frame (Reference Picture 1).						
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable
Value	Name							
1	Memory Compression Enable							
0	Memory Compression Disable							
0	Memory Compression Enable for AV1 Intra Frame This bit is for AV1 Intra Frame (Reference Picture 0).							
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable	
Value	Name							
1	Memory Compression Enable							
0	Memory Compression Disable							

AVP_TILE_CODING

AVP_TILE_CODING			
Source:	BSpec		
Length Bias:	1		
Programming Notes			
<p>This command is used only for AV1codec. It is issued for every tile of a frame. If a frame is composed of only 1 tile, it is still being issued. Tiling and Tile Group organization in AV1 cannot be disabled, a frame minimum must have 1 tile. Currently, each batch buffer can contain only 1 tile to be processed, it cannot contain more than 1 tile or the entire tile group of tiles.</p> <p>When the tile width exceeds 4096 pixels or the tile area exceeds 4096x2304 pixels, tiling must be performed and number of tiles in such frame must be > 1. There is no mandatory tiling driven by tile height. The frame height in pixels will limit the allowed tile height in extreme situation. Hence, the AVP_TILE_CODING can be issued multiple times for decoding a frame.</p> <p>Since AVP HW pipeline is stateless, all sequence, frame and segment level states (coding parameters in all Frame Level State Commands) must be reset before sending each TILE_CODING_STATE command.</p> <p>Although tile size is specified in SuperBlock unit, the minimum tile size is actually set to be 8x8 pixels (which is the same as the minimum frame size in pixels). It can also happen to the rightmost tile column and bottommost tile row of a frame which is not divisible by the SuperBlock size - this leads to the presence of partial tile and partial SuperBlock handling.</p> <p>AV1 supports both</p> <ol style="list-style-type: none"> 1) a uniform-spacing tiling scheme (as in VP9, which is always in the form of $2^N \times 2^M$ number of tiles, for the entire frame), and 2) a non-uniform-spacing tiling scheme. Bitstream syntax elements will specify the width and height of each tile size in the frame. <p>AVP HW pipeline is a tile-based codec engine, it does not need to distinguish between these two tiling schemes. Driver will take care of the difference and details of these tiling schemes. At the end, Driver will send out one tile at a time with all the related tile information to the HW through this TILE_CODING State Command.</p> <p>In AV1, a frame is partitioned by tile row and tile column. That is, a tile boundary must go across the full frame width or the full frame height only. There is no tiling within a tile.</p> <p>For AV1, the max number of tiles per frame is set to 256 in the LEVEL definition for regular video decoding. The ext-tile (Virtual Reality mode, currently not supported) has a different tiling configuration, constraints and definition.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	Opcode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	Opcode
	26:23	Media Instruction Opcode	
		Default Value:	3h Codec/Engine Name

AVP_TILE_CODING

		Format:	Opcode
	22:16	Media Instruction Command	
		Default Value:	15h AVP_TILE_CODING
		Format:	Opcode
	15:12	Reserved	
		Format:	MBZ
	11:0	Dword Length	
		Format:	U12
		Excludes Dwords 0 & 1	
		Value	Name
		4h	[Default]
1	31:24	Tile Group ID	
		Format:	U8
		<p>It specifies the Tile Group the current tile belongs to. It is intel specific parameter, not present in the AV1 bitstream.</p> <p>Each tile is belonging to a Tile Group, and there can be multiple TGs in a frame. All TGs of a frame are lined up in raster order within the frame. Each TG is received from the bitstream as a separate TG_OBU. So, a frame can receive multiple TG_OBUs. Currently, all TG_OBUs of a frame must be received in the correct sequential order (not arbitrary order is defined yet).</p> <p>A TG can start at any tile position of a frame (e.g. at the beginning of a tile row, in the middle of a tile row, or at the end of a tile row). A TG does not break by the frame width, and can continue to the next row of tiles below. A TG can span multiple tile rows. A TG can end at any tile position of a frame(e.g. at the beginning of a tile row, in the middle of a tile row, or at the end of a tile row). A TG can be viewed as a linear chain of tiles (not necessary a 2-D rectangular/square region of a frame).</p> <p>A TG can maximum contain the whole frame of tiles, or minimum contain only a single tile of the frame. A frame can be entirely coded as a single tile in a single TG.</p> <p>Each TG is assigned an unique ID number, in raster increasing order. This numbering is intel specific, not defined in the spec. There is no jump or gap in TG ID value between two adjacent TG. The top-left most tile of a frame is always has TG ID = 0. The bottom-right most tile of a frame is always has the highest TG ID value.</p> <p>Tile Group ID is in the range [0..255]. The max value 255 is intel specific.</p> <p>Note : the max number of TG per frame is set equal to the max number of tiles allowed in a frame, since each tile can belong to a different TG.</p>	
	23:12	TG Tile Num	
		Format:	U12
		<p>Specify the Tile Number inside a Tile Group. This numbering is intel specific, not present in the AV1 bitstream.</p> <p>All tiles of a TG are numbered in sequential raster order, starting from 0. TG Tile Num is reset at each TG boundary. That is, the very first tile of a TG is always assigned a TG Tile Num of 0. The last tile of a TG has the largest Tile Num of that TG.</p>	

AVP_TILE_CODING

		<p>So, the total number of tiles in a frame = (the largest Frame Tile ID + 1) = sum for all TG [largest TG Tile Num+ 1].</p> <p>Max number of tiles in a TG is currently limited to 128 at the highest level 6.3 being defined for regular video. TG Tile Num is in the range [0..255], starting from 0. The upper 5bits are reserved for future expansion.</p>		
	11:0	<p>Frame Tile ID</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U12</td> </tr> </table> <p>All the tiles of a frame are labeled with a Frame Tile ID in a sequence raster order, starting from 0. The very first tile of a frame (top-left most corner) has Frame Tile ID = 0, and the last tile of a frame (bottom-right most corner) has the largest Frame Tile ID.</p> <p>So, the total number of tiles in a frame = (the largest Frame Tile ID + 1) = sum for all TG [largest Tile Num in a TG + 1].</p> <p>Frame Tile ID is numbered from the 2 TG syntax elements TG_START and TG_END, or their default value when no present in the bitstream.</p> <p>Max number of tiles per frame is currently limited to 128 at the highest level 6.3 being defined for regular video. But for VR Large Scale Tile, max number of tiles per tile list is 512. Frame Tile ID is in the range [0..511], starting from 0. The upper 3bits are reserved for future expansion.</p> <p>Frame Tile ID numbering is consistent with the spec definition of tile ordering in a Tile Group and in a frame.</p> <p>Frame Tile ID does not get reset to 0 at Tile Group boundary (as the TG Tile Num does). Frame Tile ID numbering is not affected by dependency setting across tiles.</p>	Format:	U12
Format:	U12			
2	31:26	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
	25:16	<p>Tile Row Position in SB Unit</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U10</td> </tr> </table> <p>Specify the row (y-) position of the current tile to be processed in a frame. The position is specified in SuperBlock unit. The SuperBlock unit is specified in the Sequence Header; it can be either 64x64 pixels or 128x128 pixels.</p> <p>For regular video, the decoded tile pixels are placed in the same tile location in the decoded output frame as the coded tile in the bitstream. But for VR Large Scale Tile, the decoded tile pixels can be placed in a different location predefined in the decoded pseudo output frame as the coded tile in the bitstream. This field is used for both regular video and VR Large Scale Tile. For VR Large Scale Tile, this field is programmed to be the coded tile position for MotionCompensation purpose, and the Output Decoded Tile Row Position field is then programmed to be the position in the decoded pseudo output frame for placing the decoded pixels of the tile,</p> <p>A Tile contains a 2D array of SB units. A Tile min. size is 1 SB unit. The Tile position is based on the top-left most corner SB unit of the Tile.</p> <p>The very first tile of a frame (top-left most corner) has the position [Tile Column Position in SB Unit , Tile Row Position in SB Unit] = [0, 0]</p> <p>The frame height is rounded up to an integer multiple of the Superblock unit.</p> <p>A frame height that is less than a SB unit (but must be >=8 pixels), is rounded up to 1 SB unit for the purpose of defining a tile.</p>	Format:	U10
Format:	U10			

AVP_TILE_CODING

A frame height that is not divisible by tile height, the last tile row of the frame will have a smaller tile height, but still an integer multiple of SB unit. It does not affect the Tile Row Position in SB Unit of these partial Tiles.

Tile Row Position in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()).

Max Frame Height = 64K, hence the max number of SB unit= 1024, when the SB unit is 64x64 pixels, and the max Tile Row Position in SB unit can be 1023.

Intel supports Tile Row Position in SB unit only in the range of [0..255], for 16K Max Frame Height. starting from 0 at the top-left most corner of a frame.

Value	Name	Description
[0,255]	16K_Below	This allows support up to 16K picture

15:10 **Reserved**

Format:	MBZ
---------	-----

9:0 **Tile Column Position in SB Unit**

Format:	U10
---------	-----

Specify the column (x-) position of the current tile to be processed in a frame. The position is specified in SuperBlock unit. The SuperBlock unit is specified in the Sequence Header; it can be either 64x64 pixels or 128x128 pixels.

For regular video, the decoded tile pixels are placed in the same tile location in the decoded output frame as the coded tile in the bitstream. But for VR Large Scale Tile, the decoded tile pixels can be placed in a different location predefined in the decoded pseudo output frame as the coded tile in the bitstream. This field is used for both regular video and VR Large Scale Tile. For VR Large Scale Tile, this field is programmed to be the coded tile position for MotionCompensation purpose, and the Output Decoded Tile Column Position field is then programmed to be the position in the decoded pseudo output frame for placing the decoded pixels of the tile,

A Tile contains a 2D array of SB units. A Tile min. size is 1 SB unit. The Tile position is based on the top-left most corner SB unit of the Tile.

The very first tile of a frame (top-left most corner) has the position [Tile Column Position in SB Unit , Tile Row Position in SB Unit] = [0, 0]

The frame width is rounded up to an integer multiple of the Superblock unit.

A frame width that is less than a SB unit (but must be ≥ 8 pixels), is rounded up to 1 SB unit for the purpose of defining a tile.

A frame width that is not divisible by tile width, the last tile column of the frame will have a smaller tile width, but still an integer multiple of SB unit. It does not affect the Tile Column Position in SB Unit of these partial Tiles.

Tile Column Position in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()).

Max Frame Width= 64K, hence the max number of SB unit= 1024, when the SB unit is 64x64 pixels, and the max Column Row Position in SB unit can be 1023.

Intel supports Tile Column Position in SB unit only in the range of [0..255], for 16K Max Frame Width. starting from 0 at the top-left most corner of a frame.

Value	Name	Description
[0,255]	16K_Below	This allows support up to 16K picture.

AVP_TILE_CODING

3	31:26	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									
	25:16	Tile Height in SuperBlock Unit Minus1 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U10</td> </tr> </table> <p>Tile Size is measured in SuperBlock Unit. The SuperBlock unit is specified in the Sequence Header; it can be either 64x64 pixels or 128x128 pixels.</p> <p>When large-scale tile is ON, tile height must be 1 SB in size (i.e. can be 64x64 or 128x128, depending on the SB size flag).</p> <p>The minimum Tile Size is 1 SB unit x 1 SB unit</p> <p>The frame height is rounded up to an integer multiple of the Superblock unit. If a frame height is not divisible by the SuperBlock unit, the bottommost row of SBs of the frame is partial in size, but for the purpose of tile size definition, the partial SB is still counted as 1 unit.</p> <p>A frame height that is less than a SB unit, is rounded up to 1 SB unit for the purpose of defining a tile.</p> <p>A frame height that is not divisible by tile height, the last tile row of the frame will have a smaller tile height, but still an integer multiple of SB unit.</p> <p>Tile Height in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()).</p> <p>In AV1, there are two max tile size constraints : Max Tile Width <= 4096 pixels and Max Tile Area <= 4096 width x2304 height pixels. But there is no separate constraint for Max Tile Height.</p> <p>Intel set a limit for frame height to be 16K pixels.</p> <p>In AV1, the following restrictions apply:</p> <ol style="list-style-type: none"> 1) Last SB (if partial in size)at frames bottommost edge must align to 8x8 block (partial SB) 2) In Scalability Mode, the minimum tile size is 2 width x1 height SBs. (Intel restriction) <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,255]</td> <td>16K_Below</td> <td>This supports up to 16k height tiles</td> </tr> </tbody> </table>	Format:	U10	Value	Name	Description	[0,255]	16K_Below	This supports up to 16k height tiles
Format:	U10									
Value	Name	Description								
[0,255]	16K_Below	This supports up to 16k height tiles								
	15:6	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									
	5:0	Tile Width in SuperBlock Unit Minus1 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U6</td> </tr> </table> <p>Tile Size is measured in SuperBlock Unit.The SuperBlock unit is specified in the Sequence Header; it can be either 64x64 pixels or 128x128 pixels.</p> <p>The minimum Tile Size is 1 SB unit x 1 SB unit</p> <p>The frame width is rounded up to an integer multiple of the Superblock unit. If a width height is not divisible by the SuperBlock unit, the rightmost column of SBs of the frame is partial in size, but for the purpose of tile size definition, the partial SB is still counted as 1 unit.</p> <p>A frame width that is less than a SB unit, is rounded up to 1 SB unit for the purpose of defining a tile.</p> <p>A frame width that is not divisible by tile width, the last tile column of the frame will have a smaller tile width, but still an integer multiple of SB unit.</p> <p>Tile Width in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()).</p> <p>In AV1, there are two max tile size constraints : Max Tile Width <= 4096 pixels and Max Tile Area <= 4096 width x2304 height pixels. But there is no separate constraint for Max Tile Height. When</p>	Format:	U6						
Format:	U6									

AVP_TILE_CODING

		<p>super-res is ON, these tile constraints are applied to the downscaled frame's tiles. So, Tile Width in SuperBlock Unit Minus1 is in the range of [0..63]. Tile Width = (Tile Width in SuperBlock Unit Minus1 + 1) SBs. In AV1, the following restrictions apply. 1) Last SB (if partial in size)at frames rightmost edge must align to 8x8 block (partial SB) 2) In Scalability Mode, the minimum tile size is 2 width x1 height SBs. (Intel restriction)</p>		
4	31	<p>Disable Frame Context Update Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Set to 0, the current tile being decoded is writing out its updated Frame Context to memory (surface buffer), at the end of its decoding. Set to 1, the current tile being decoded is not writing out its updated Frame Context to memory, for use in the next frame to be decoded. It is the frame level syntax element, disable_frame_end_update_cdf, or named as [!refresh_frame_context]. Default is 0. The arithmetic decoding of each tile of a frame is started with the same Frame Context (CDF Cumulative Probability Table Set for all AV1 Syntax Elements) provided by the Driver. This is from a read-only memory surface. After each arithmetic decoding of a Syntax Element from the bitstream, its corresponding CDF Table will be updated (if Disable CDF Update Flag = 0). As such, the frame context is changed after decoding each tile. When Backward Adaptation of the current frame context (CDF Cumulative Probability Table Set for all AV1 Syntax Elements) is enabled, only the largest tile in byte size (bitstream size) of the current frame being decoded will update the frame context for the decoding of the next frame. Driver will determine which tile in the current frame will need to write out its updated frame context to memory by setting Disable Frame Context Update Flag to 0. This is to a write-only memory surface. For all other tiles of the frame, Driver will set Disable Frame Context Update Flag to 1. If Driver knows, ahead of time, which tile has the largest bitstream size, then the frame context update can only be done once. If Driver does not know, ahead of time, which tile has the largest bitstream size, then the frame context update may be done more than once, and each time overwritten the previous one, until the largest tile has found. When in Intel Scalability mode (multiple HW pipes), there are still be one read-only and one write-only frame context buffers allocated for decoding the current frame. And at anyone time, Driver will only enable one pipe to update/write out the frame context. Again, it is possible that this update can be done more than once until the largest tile among all pipes has found. Note: Even when Error Resilient Mode is ON, this field can still be 0 or 1.</p>	Format:	U1
	Format:	U1		
30	<p>Disable CDF Update Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Set to 1, the current tile being decoded is not updating the CDF table of each syntax element after parsing(multisymbol arithmetic decode) from the bitstream. Hence, the frame context is not being changed. And no need to write out the frame context at the end of decoding the frame. Set to 0, the current tile being decoded is updating the CDF table of each syntax element after parsing (multisymbol arithmetic decode) from the bitstream. Hence. the frame context is</p>	Format:	U1	
Format:	U1			

AVP_TILE_CODING

	<p>changed. If Disable Frame Context Update Flag = 0 for this tile, the new frame context is writing out to memory for subsequent frame decoding. It is the frame level syntax element, disable_cdf_update. Default is 0.</p>		
29	<p>IsLastTileOfFrame Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U1</td> </tr> </table> <p>Indicates if current tile being decoded is the last tile of a frame. Default is 0. The last tile is the bottom-right most corner region of a frame. This is intel specific frame level parameter.</p>	Format:	U1
Format:	U1		
28	<p>IsEndTileOfTileGroup Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U1</td> </tr> </table> <p>Indicates if current tile is the last tile of a Tile Group. Default is 0. 0 - is not the end tile of a tile group 1 - is the end tile of a tile group. It is derived as : isEndTileOfTileGroup = (Frame Tile ID = = TG_END syntax element) A Tile Group can end at any tile in a tile row (e.g. the first tile of a tile row, the last tile of a tile row, or a tile anywhere in the middle of a tile row). The End tile of a Tile Group may not be in the same tile row as the Start tile of the same Tile Group, but must come after in raster order. This is intel specific frame level parameter.</p>	Format:	U1
Format:	U1		
27	<p>IsStartTileOfTileGroup Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U1</td> </tr> </table> <p>Indicates if current tile is the first tile of a Tile Group. Default is 0. 0 - is not the start tile of a tile group 1 - is the start tile of a tile group. It is derived as : isStartTileOfTileGroup = (Frame Tile ID = = TG_START syntax element) A Tile Group can start at any tile in a tile row (e.g. the first tile of a tile row, the last tile of a tile row, or a tile anywhere in the middle of a tile row). The End tile of a Tile Group may not be in the same tile row as the Start tile of the same Tile Group. This is intel specific frame level parameter.</p>	Format:	U1
Format:	U1		
26	<p>IsLastTileOfRow Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U1</td> </tr> </table> <p>Indicates if the current tile is the last tile of the current tile row. Default is 0. 0 - is not the last tile of a tile row 1 - is the last tile of a tile row. It is derived from Frame Tile ID and Num of Tile Columns Minus1. It is the tile at the right frame boundary. The bottom-right most tile of a frame is having both IsLastTileOfRow and IsLastTileOfColumn set to 1. This is intel specific frame level parameter.</p>	Format:	U1
Format:	U1		

AVP_TILE_CODING

AVP_TILE_CODING			
	25	IsLastTileOfColumn Flag	Format: U1
	<p>Indicates if the current tile is the last tile of the current tile column. Default is 0.</p> <p>0 - is not the last tile of a tile column</p> <p>1 - is the last tile of a tile column.</p> <p>It is derived from Frame Tile ID, Num of Tile Columns Minus1 and Num of Tile Rows Minus1. is the tile at the bottom frame boundary.</p> <p>The bottom-right most tile of a frame is having both IsLastTileOfRow and IsLastTileOfColumn set to 1.</p> <p>This is intel specific frame level parameter.</p>		
	24	Reserved	Format: MBZ
	23	Reserved	Format: MBZ
	22:4	Reserved	Format: MBZ
	3:0	Reserved	Format: MBZ
5	31:22	Num of Tile Rows Minus1 in a Frame	Format: U10
	<p>Specify the total number of Tile Rows in a frame.</p> <p>$\text{TileRows} = \text{Num of Tile Rows Minus1 in a Frame} + 1$.</p> <p>Max Frame Height is 64K, the smallest tile size is 1SB unit, hence max number of tile rows in a frame can be 1024.</p> <p>Valid Num of Tile Rows Minus1 in a Frame is in the range of [0..63] only. The max tile number in a frame is governed by the Level definition for a compliant bitstream.</p> <p>For regular video, highest Level defined is 6.3 (max. 8K video), which specifies max total 128 tiles in a frame and max number of tile columns is 16. The corresponding tile configuration is flexible, it can be 16x8, 8x16, 16x4, 4x16, 8x8, etc. There is no constraints on max number of tile rows.</p> <p>For VR large scale tile application, the tile configuration can be max total 64x64=4K tiles in a pseudo output frame and an anchor frame. But the tile list can max. contain 512 tiles at a time. This is the same as the variable tile_rows (minus1) defined in the reference C model.</p> <p>Num of Tile Rows in a frame is derived from the frame level syntax elements in tiling specifications (tile_info()).</p>		
	21:12	Num of Tile Columns Minus1 in a Frame	Format: U10
<p>Specify the total number of Tile Columns in a frame.</p> <p>$\text{TileColumns} = \text{Num of Tile Columns Minus1 in a Frame} + 1$.</p> <p>Max Frame Width is 64K, the smallest tile size is 1SB unit, hence max number of tile columns in a</p>			

AVP_TILE_CODING

	<p>frame can be 1024.</p> <p>Valid Num of Tile Columns Minus1 in a Frame is in the range of [0..63] only. The max tile number in a frame is governed by the Level definition for a compliant bitstream.</p> <p>For regular video, highest Level defined is 6.3 (max. 8K video), which specifies max total 128 tiles in a frame and max number of tile columns is 16. The corresponding tile configuration is flexible, it can be 16x8, 8x16, 16x4, 4x16, 8x8, etc. There is no constraints on max number of tile rows.</p> <p>For VR large scale tile application, the tile configuration can be max total 64x64=4K tiles in a pseudo output frame and an anchor frame. But the tile list can max. contain 512 tiles at a time. This is the same as the variable tile_cols (minus1) defined in the reference C model.</p> <p>Num of Tile Columns in a frame is derived from the frame level syntax elements in tiling specifications (tile_info()).</p> <p>In Intel Scalability Mode (multiple HW pipes), execution across multiple pipes is done in tile column fashion. When the number of tile columns in a frame > the number of HW pipes, the tile columns are cycled through the pipes in multiple phases as described below:</p> <p>Tile Col0/Row0 Tile Col1/Row0 Tile Col2/Row0 Tile Col0/Row1 Tile Col1/Row1 Tile Col2/Row1 Tile Col0/Row2 Tile Col1/Row2 Tile Col2/Row2</p> <p>Assume there is only 2 HW pipes (Pipe0 and Pipe1)</p> <p>Phase 0: Pipe0 :Tile Col0/Row0 ; Pipe1 : Tile Col1/Row0</p> <p>Phase 1: Pipe0 :Tile Col2/Row0</p> <p>Phase 2: Pipe0 :Tile Col0/Row1 ; Pipe1 : Tile Col1/Row1</p> <p>Phase 3: Pipe0 :Tile Col2/Row1</p> <p>Phase 4: Pipe0 :Tile Col0/Row2 ; Pipe1 : Tile Col1/Row2</p> <p>Phase 5: Pipe0 :Tile Col2/Row2.</p>							
11:8	<p>Reserved MBZ</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>		Format:	MBZ				
Format:	MBZ							
7:0	<p>Number of Active BE Pipes</p> <p>Indicates the number of active, consecutive positioned Scalable VDBOXs to be used for the current frame decoding or encoding.</p> <p>BE Pipe partitioning, SW must guarantee the minimum width is at least two full SBs for each tiles</p> <p>This field in general should be smaller or equal to Num of Tile columns in a Frame.</p> <p>This field is ignored by HW</p> <p>This field is not used by HW</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 45%;">Comment</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>ignored</td> </tr> <tr> <td style="text-align: center;">1</td> <td>ignored</td> </tr> </tbody> </table>		Value	Comment	0	ignored	1	ignored
Value	Comment							
0	ignored							
1	ignored							

AVP_VD_CONTROL_STATE

AVP_VD_CONTROL_STATE			
Source:	VideoCS		
Length Bias:	2		
For AVP, it is selected with the Media Instruction Opcode "3h" .			
This command is used to modify the control of HCP pipe. It can be inserted anywhere within a frame. It can be inserted multiple times within a frame as well.			
This command is also used for AVP pipe.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
Default Value:		3h Codec/Engine Name for AVP	
Format:		OpCode	
Codec/EngineName = AVP = 3h			
22:16	Media Instruction Command		
	Default Value:	Ah VD_CONTROL_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
2h			
1..2	63:0	VD Control State Body	
		Format:	VD_CONTROL_STATE_BODY

Barrier

MSD_BARRIER - Barrier											
Source:	EuSubFunctionGateway										
Length Bias:	1										
Record an additional thread reaching the barrier.											
DWord	Bit	Description									
0	31:29	Reserved Format: _____ MBZ									
	28:25	Message Length Format: _____ U4 Specifies the number of GRF registers sent as the message payload. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>One [Default]</td> <td>See MDP_Barrier Barrier Data Payload definition.</td> </tr> </tbody> </table>	Value	Name	Description	1	One [Default]	See MDP_Barrier Barrier Data Payload definition.			
	Value	Name	Description								
	1	One [Default]	See MDP_Barrier Barrier Data Payload definition.								
	24:20	Response Length Format: _____ U5 Specifies the number of GRF registers expected as the message response payload. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>None [Default]</td> <td>Barrier completion notification is signaled with ARF N0.0.</td> </tr> <tr> <td>1</td> <td>One</td> <td>Ignore the data in the register. The update of the register is used as the barrier completion notification</td> </tr> </tbody> </table>	Value	Name	Description	0	None [Default]	Barrier completion notification is signaled with ARF N0.0.	1	One	Ignore the data in the register. The update of the register is used as the barrier completion notification
	Value	Name	Description								
	0	None [Default]	Barrier completion notification is signaled with ARF N0.0.								
	1	One	Ignore the data in the register. The update of the register is used as the barrier completion notification								
	19:16	Reserved Format: _____ MBZ									
	15:12	Fence Data Ports Format: _____ GW_FENCE_PORTS Bit mask specifies the list of data ports to be fenced with this barrier message. <table border="1"> <thead> <tr> <th>Restriction</th> </tr> </thead> <tbody> <tr> <td>Ignored.</td> </tr> </tbody> </table>	Restriction	Ignored.							
Restriction											
Ignored.											
11	Active Threads Only Format: _____ Enable If this field is set, only the "active" threads (i.e. threads that have not EOT-ed) in thread-group is counted toward the barrier completion. Also, barrier type is forced to "Producer-Consumer", ignoring the setting in the barrier payload.										
10:3	Reserved Format: _____ MBZ										
2:0	Barrier Subfunction										

MSD_BARRIER - Barrier

		Default Value:	0x4
		Format:	OpCode

Bit Field Extract

bfe - Bit Field Extract			
Source:	Eulsa		
Length Bias:	4		
<p>Component-wise extract a bit field from src2 using the bit field width from src0 and the bit field offset from src1. Store the extracted bit field value in the low bits of dst and sign extend (if D type) or zero extend (if UD type). The width and offset values are from the low five bits of src0 and src1 respectively, or src0 & 0x1f and src1 & 0x1f. If width is zero, the result is zero. If offset + width > 32 then the extracted bit field is bits offset to 31 of src2, extracting only 32 - offset bits, less than width as the bit field cannot extend past the MSB of the source value. Otherwise extract width bits extending from bit positions offset to offset + width - 1.</p>			
Format:	<pre>[(pred)] bfe (exec_size) dst src0 src1 src2</pre>		
Restriction			
No accumulator access, implicit or explicit.			
All three-source instructions have certain restrictions, described in Instruction Formats.			
Syntax			
<pre>[(pred)] bfe (exec_size) reg reg reg reg</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD width = src0.chan[n][4:0]; UD offset = src1.chan[n][4:0]; if (width == 0) { dst.chan[n] = 0x00000000; } else if ((width + offset) < 32) { dst.chan[n] = src2.chan[n] << (32 - width - offset); if (src2 is signed) { dst.chan[n] = dst.chan[n] >> (32 - width); // pad sign bit of dst.chan } else { dst.chan[n] = dst.chan[n] >> (32 - width); // pad 0 } } else if (if (src2 is signed) { dst.chan[n] = src2.chan[n] >> offset; // pad sign bit } else { dst.chan[n] = src2.chan[n] >> offset; // pad 0 } } } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
Src Types	Dst Types		
UD	UD		
D	D		
DWord	Bit	Description	
0..3	127:114	Src2.Operand	
		Exists If:	([Src2.IsLmm]==false) AND ([Header][Opcode]!=madm)
		Format:	DirectOperand
	127:114	Src2.Operand	

bfe - Bit Field Extract

	Exists If:	([Src2.IsImm]==false) AND ([Header][Opcode]==madm)	
	Format:	MacroOperand	
127:112	Src2.ImmValue[15:0]		
	Exists If:	([Src2.IsImm]==true)	
113:112	Src2.HorzStride		
	Exists If:	([Src2.IsImm]==false)	
	Format:	HorzStride	
111:98	Src1.Operand		
	Exists If:	([Header][Opcode]!=madm)	
	Format:	DirectOperand	
111:98	Src1.Operand		
	Exists If:	([Header][Opcode]==madm)	
	Format:	MacroOperand	
97:96	Src1.HorzStride		
	Format:	HorzStride	
95:92	CondCtrl		
	Format:	FlagModifier	
91	Src1.VertStride[1]		
	Format:	TernaryVertStride[1]	
90:88	Src1.DataType		
	Format:	TernaryDataType	
87:86	Src1.Mod		
	Format:	SrcMod	
85:84	Src2.Mod		
	Format:	SrcMod	
83	Src1.VertStride[0]		
	Format:	TernaryVertStride[0]	
82:80	Src2.DataType		
	Format:	TernaryDataType	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]!=madm)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]==madm)	
	Format:	MacroOperand	
79:64	Src0.ImmValue[15:0]		

bfe - Bit Field Extract

		Exists If:	([Src0.IsImm]==true)
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm]==false)	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Header][Opcode]!=madm)	
	Format:	DirectOperand	
	Programming Notes		
The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,			
63:50	Dst.Operand		
	Exists If:	([Header][Opcode]==madm)	
	Format:	MacroOperand	
49	Reserved		
	Format:	MBZ	
48	Dst.HorzStride		
	This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand.		
	Value	Name	
	0	1 element	
1	2 element		
47	Src2.IsImm		
	This field indicate that Source 2 operand is carrying an immediate value.		
	Value	Name	
	0	false	
1	true		
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
1	true		
45:44	Src0.Mod		
	Format:	SrcMod	
43	Src0.VertStride[1]		
	Format:	TernaryVertStride[1]	
42:40	Src0.DataType		
	Format:	TernaryDataType	

bfe - Bit Field Extract

39	ExecDataType This field indicate the datatype mode of ternary instruction. Integer or Float. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Integer</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Float</td> </tr> </tbody> </table>	Value	Name	0	Integer	1	Float			
Value	Name									
0	Integer									
1	Float									
38:36	Dst.DataType Format: TernaryDataType									
35	Src0.VertStride[0] Format: TernaryVertStride[0]									
34	Saturate Format: Saturate									
33	AccWrCtrl Format: AccWrCtrl									
32	AtomicCtrl Format: AtomicCtrl									
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description								
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.								
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.								
30	Reserved									
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.								
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask									

bfe - Bit Field Extract

	<p>for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize							
Format:	ExecSize									
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>	Format:	Header							
Format:	Header									

Bit Field Insert 1

bfi1 - Bit Field Insert 1			
Source:	Eulsa		
Length Bias:	4		
<p>The bfi1 instruction is the first instruction in a two-instruction macro for bfi (Bit Field Insert). The bfi1 instruction component-wise generates mask with control from src0 and src1 and stores the results in dst. The mask is used in the bfi2 instruction to generate the final result of bfi. Create a bit mask corresponding to the bit field width and offset in src0 and src1. Store the bit mask in dst. The mask has all bits in the bit field set to 1 and all other bits as 0. The width and offset values are from the low five bits of src0 and src1 respectively, or src0 & 0x1f and src1 & 0x1f. If width is zero, the result is zero. The bfi macro has four source operands: src0 - bit field width in low five bits, src1 - bit field offset/starting bit position in low five bits, src2 - bit field value to insert, using only the number of least significant bits given by width in src0, and src3 - overall value into which the bit field is inserted, providing all bits other than the inserted bits for the result value. bfi dst src0 src1 src2 src3 //</p> <p>Translates to these two instructions: bfi1 dst src0 src1 bfi2 dst dst src2 src3</p>			
<p>Format:</p> <pre>[(pred)] bfi1 (exec_size) dst src0 src1</pre>			
Programming Notes			
No accumulator access, implicit or explicit.			
Syntax			
<pre>[(pred)] bfi1 (exec_size) reg reg reg [(pred)] bfi1 (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate (WrEn);) { src1.chan[n][4:0]; } for (n = 0; n < exec_size; n++) { UD width = src0.chan[n][4:0]; dst = ((1 << width) - 1) << offset; if (WrEn.chan[n] UD offset = }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
Src Types	Dst Types		
UD	UD		
D	D		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)

bfi1 - Bit Field Insert 1

	125:122	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	121:120	Src1.Mod	
		Exists If:	([Src1.IsImm]==false)
		Format:	SrcMod
	119:116	Src1.VertStride	
		Exists If:	([Src1.IsImm]==false)
		Format:	VertStride
	115:113	Src1.Width	
		Exists If:	([Src1.IsImm]==false)
		Format:	Width
	112	Src1.AddrMode	
		Exists If:	([Src1.IsImm]==false)
	Format:	AddrMode	
111:98	Src1.Operand		
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)	
	Format:	DirectOperand	
111:98	Src1.Operand		
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)	
	Format:	IndirectOperand	
97:96	Src1.HorzStride		
	Exists If:	([Src1.IsImm]==false)	
	Format:	HorzStride	
95:92	CondCtrl		
	Format:	FlagModifier	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==false)	
	Format:	RegDataType	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==true)	
	Format:	ImmDataType	
87:84	Src0.VertStride		
	Format:	VertStride	
83:81	Src0.Width		
	Format:	Width	

bfi1 - Bit Field Insert 1

80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
1	true	
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
1	true	
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==true)
	Format:	ImmDataType

bfi1 - Bit Field Insert 1

39:36	Dst.DataType	Format: RegDataType												
35	Dst.AddrMode	Format: AddrMode												
34	Saturate	Format: Saturate												
33	AccWrCtrl	Format: AccWrCtrl												
32	AtomicCtrl	Format: AtomicCtrl												
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 65%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>		Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.		
Value	Name	Description												
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.												
1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.												
30	Reserved													
29	CmptCtrl Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;">Format:</td> <td style="width: 85%; text-align: right;">MBZ</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 65%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>		Format:	MBZ	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Format:	MBZ													
Value	Name	Description												
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.												
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.												
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 65%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Positive</td> <td>Positive polarity of predication. Use the predication mask produced</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive	Positive polarity of predication. Use the predication mask produced					
Value	Name	Description												
0	Positive	Positive polarity of predication. Use the predication mask produced												

bfi1 - Bit Field Insert 1

		[Default]	by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff Format: ChanOff This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.		
18:16	ExecSize Format: ExecSize This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.		
15:0	Header Format: Header		

Bit Field Insert 2

bfi2 - Bit Field Insert 2			
Source:	Eulsa		
Length Bias:	4		
<p>The bfi2 instruction is the second instruction in a two-instruction macro for bfi (Bit Field Insert). The bfi2 instruction component-wise performs the bitfield insert operation on src1 and src2 based on the mask in src0. Use the mask in src0 to take a bit field value from the low bits of src1 and combine it with the value from src2 (so src2 provides all bits other than those masked out and replaced by the bit field value). Store the result in dst. The bfi macro has four source operands: src0 - bit field width in low five bits, src1 - bit field offset/starting bit position in low five bits, src2 - bit field value to insert, using only the number of least significant bits given by width in src0, and src3 - overall value into which the bit field is inserted, providing all bits other than the inserted bits for the result value. bfi dst src0 src1 src2 src3 // Translates to these two instructions: bfi1 dst src0 src1 bfi2 dst dst src2 src3</p>			
Format:	<pre>[(pred)] bfi2 (exec_size) dst src0 src1 src2</pre>		
Restriction			
No accumulator access, implicit or explicit.			
All three-source instructions have certain restrictions, described in Instruction Formats.			
Syntax			
<pre>[(pred)] bfi2 (exec_size) reg reg reg reg</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD offset = LZD(reverse(src0.chan[n]))-1; // offset is the number of LSB zero bits below the bit mask which has all 1s. // width (implied by the logic) is the number of 1 bits in the mask value, which should be all 1s. dst.chan[n] = ((src1.chan[n] << offset) & src0.chan[n]) (src2.chan[n] & ! src0.chan[n]); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
Src Types	Dst Types		
UD	UD		
D	D		
DWord	Bit	Description	
0..3	127:114	Src2.Operand	
		Exists If:	([Src2.IsImm]==false) AND ([Header][Opcode]!=madm)
		Format:	DirectOperand
	127:114	Src2.Operand	

bfi2 - Bit Field Insert 2

	Exists If:	([Src2.IsImm]==false) AND ([Header][Opcode]==madm)	
	Format:	MacroOperand	
127:112	Src2.ImmValue[15:0]		
	Exists If:	([Src2.IsImm]==true)	
113:112	Src2.HorzStride		
	Exists If:	([Src2.IsImm]==false)	
	Format:	HorzStride	
111:98	Src1.Operand		
	Exists If:	([Header][Opcode]!=madm)	
	Format:	DirectOperand	
111:98	Src1.Operand		
	Exists If:	([Header][Opcode]==madm)	
	Format:	MacroOperand	
97:96	Src1.HorzStride		
	Format:	HorzStride	
95:92	CondCtrl		
	Format:	FlagModifier	
91	Src1.VertStride[1]		
	Format:	TernaryVertStride[1]	
90:88	Src1.DataType		
	Format:	TernaryDataType	
87:86	Src1.Mod		
	Format:	SrcMod	
85:84	Src2.Mod		
	Format:	SrcMod	
83	Src1.VertStride[0]		
	Format:	TernaryVertStride[0]	
82:80	Src2.DataType		
	Format:	TernaryDataType	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]!=madm)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]==madm)	
	Format:	MacroOperand	
79:64	Src0.ImmValue[15:0]		

bfi2 - Bit Field Insert 2

		Exists If:	([Src0.IsImm]==true)
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm]==false)	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Header][Opcode]!=madm)	
	Format:	DirectOperand	
	Programming Notes		
The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,			
63:50	Dst.Operand		
	Exists If:	([Header][Opcode]==madm)	
	Format:	MacroOperand	
49	Reserved		
	Format:	MBZ	
48	Dst.HorzStride		
	This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand.		
	Value	Name	
	0	1 element	
1	2 element		
47	Src2.IsImm		
	This field indicate that Source 2 operand is carrying an immediate value.		
	Value	Name	
	0	false	
1	true		
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
1	true		
45:44	Src0.Mod		
	Format:	SrcMod	
43	Src0.VertStride[1]		
	Format:	TernaryVertStride[1]	
42:40	Src0.DataType		
	Format:	TernaryDataType	

bfi2 - Bit Field Insert 2

39	ExecDataType This field indicate the datatype mode of ternary instruction. Integer or Float. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Integer</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Float</td> </tr> </tbody> </table>	Value	Name	0	Integer	1	Float			
Value	Name									
0	Integer									
1	Float									
38:36	Dst.DataType Format: TernaryDataType									
35	Src0.VertStride[0] Format: TernaryVertStride[0]									
34	Saturate Format: Saturate									
33	AccWrCtrl Format: AccWrCtrl									
32	AtomicCtrl Format: AtomicCtrl									
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%; text-align: center;">Value</th> <th style="width: 15%; text-align: center;">Name</th> <th style="width: 70%; text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">NoMask</td> <td>NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description								
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.								
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.								
30	Reserved									
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%; text-align: center;">Value</th> <th style="width: 15%; text-align: center;">Name</th> <th style="width: 70%; text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.								
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask									

bfi2 - Bit Field Insert 2

	<p>for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. This can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize							
Format:	ExecSize									
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>	Format:	Header							
Format:	Header									

Bit Field Reverse

bfrev - Bit Field Reverse			
Source:	Eulsa		
Length Bias:	4		
The bfrev instruction component-wise reverses all the bits in src0 and stores the results in dst.			
Format:	[(pred)] bfrev (exec_size) dst src0		
Restriction			
No accumulator access, implicit or explicit.			
Syntax			
[(pred)] bfrev (exec_size) reg reg [(pred)] bfrev (exec_size) reg imm32			
Pseudocode			
<pre>Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { for (idx = 0; idx < 32; idx++) { dst.chan[n][idx] = src0.chan[n][31-idx]; } } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
Src Types	Dst Types		
UD	UD		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0] Exists If: $[(Src0.IsImm) == true]$	
	95:92	CondCtrl Exists If: $[(Src0.IsImm) == false] \text{ OR } (((Src0.DataType) != :q) \text{ AND } [(Src0.DataType) != :uq] \text{ AND } [(Src0.DataType) != :df])$ Format: FlagModifier	
	95:64	Src0.ImmValue[63:32] Exists If: $[(Src0.IsImm) == true] \text{ AND } (((Src0.DataType) == :q) \text{ OR } [(Src0.DataType) == :uq] \text{ OR } [(Src0.DataType) == :df])$	
	87:84	Src0.VertStride Exists If: $[(Src0.IsImm) == false] \text{ OR } (((Src0.DataType) != :q) \text{ AND } [(Src0.DataType) != :uq] \text{ AND } [(Src0.DataType) != :df])$ Format: VertStride	
	83:81	Src0.Width	

bfrev - Bit Field Reverse

	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	Width	
80	Src0.AddrMode		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		

bfrev - Bit Field Reverse

		Exists If:	([Src0.IsImm]==false)
		Format:	RegDataType
43:40	Src0.DataType		
		Exists If:	([Src0.IsImm]==true)
		Format:	ImmDataType
39:36	Dst.DataType		
		Format:	RegDataType
35	Dst.AddrMode		
		Format:	AddrMode
34	Saturate		
		Format:	Saturate
33	AccWrCtrl		
		Format:	AccWrCtrl
32	AtomicCtrl		
		Format:	AtomicCtrl
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
		Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask		

bfrev - Bit Field Reverse

		<p>for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 65%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	<p>PredCtrl</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl								
Format:	PredCtrl										
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>										
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>										
21:19	<p>ChanOff</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff								
Format:	ChanOff										
18:16	<p>ExecSize</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize								
Format:	ExecSize										
15:0	<p>Header</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">Header</td> </tr> </table>	Format:	Header								
Format:	Header										

Branch Converging

brc - Branch Converging			
Source:	Eulsa		
Length Bias:	4		
<p>The brc instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if all channels are branched away. UIP should reference the instruction where all channels are expected to come together. JIP should reference the end of the innermost conditional block.</p> <p>In instruction binary, JIP and UIP use locations src1 and src0 respectively when immediate and location src0 when reg64, where reg64 is accessed as paired DWord (regioning being <2;2,1>). dst must be IP. When the offsets are immediate, src0 regfile must be immediate.</p>			
Format:	[(pred)] brc (exec_size) JIP UIP		
Restriction			
A brc instruction cannot use the Switch instruction option.			
Syntax			
[(pred)] brc (exec_size) imm32 imm32 [(pred)] brc (exec_size) reg64			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn[n]) { PcIP[n] = IP + UIP; } else { PcIP[n] = IP + 1; } } if (all PcIP != IP + 1) { // for all channels Jump(IP + JIP); }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
	Format:	MBZ	
	127:96	JIP	
Exists If:		([Src0.IsImm]==true)	
Format:		S31	
		The byte-aligned jump distance if a jump is taken for the channel.	
95:80	Reserved		
	Exists If:	([Src0.IsImm]==false)	
Format:	MBZ		
95:64	Reserved		

brc - Branch Converging

	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)	
	Format:	MBZ	
95:64	UIP		
	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)	
	Format:	S31	
	The byte aligned jump distance if a jump is taken for the instruction.		
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false)	
	Format:	DirectOperand	
65:64	Reserved		
	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
63:50	Dst.Operand		
	Format:	DirectOperand	
49:48	Reserved		
	Format:	MBZ	
47	Src1.IsImm		
	This field indicate that Source 1 operand is carrying an immediate value		
	Value	Name	
	0	false	
	1	true	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value		
	Value	Name	
	0	false	
	1	true	
45:34	Reserved		
	Format:	MBZ	
33	BranchCtrl		
	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.		
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		

brc - Branch Converging

Value	Name	Description
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
Format:		MBZ
<p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p>		
Value	Name	Description
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
<p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p>		
Value	Name	Description
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl	
Format:		PredCtrl
<p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		
23	FlagRegNum[0]	
<p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	FlagSubRegNum	
<p>This field specifies the sub-register number for a flag register operand. There are two sub-</p>		

brc - Branch Converging			
	<p>registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Branch Diverging

brd - Branch Diverging				
Source:	Eulsa			
Length Bias:	4			
<p>The brd instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if any channels are branched away.</p> <p>In instruction binary, JIP is at location src1 when immediate and at location src0 when reg32, where reg32 is accessed as a scalar DWord. The ip register must be used (for example, by the assembler) as dst.</p>				
Format:	[(pred)] brd (exec_size) JIP			
Restriction				
A brd instruction cannot use the Switch instruction option.				
Syntax				
[(pred)] brd (exec_size) imm32		[(pred)] brd (exec_size) reg32		
Pseudocode				
<pre> Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn[n]) { PcIP[n] = IP + JIP; } else { PcIP[n] = IP + 1; } } if (any PcIP == ExIP + JIP) { // any channel Jump(ExIP + JIP); } </pre>				
Predication	Conditional Modifier	Saturation	Source Modifier	
Y	N	N	N	
DWord	Bit	Description		
0.3	127:96	Reserved		
		Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ		
	127:96	JIP		
		Exists If:	([Src0.IsImm]==true)	
		Format:	S31	
			The byte-aligned jump distance if a jump is taken for the channel	
	95:80	Reserved		
Exists If:		([Src0.IsImm]==false)		
Format:	MBZ			
95:64	Reserved			
	Exists If:	([Src0.IsImm]==true)		

brd - Branch Diverging

	Format:	MBZ	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false)	
	Format:	DirectOperand	
65:64	Reserved		
	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
63:50	Dst.Operand		
	Format:	DirectOperand	
49:47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
1	true		
45:34	Reserved		
	Format:	MBZ	
33	BranchCtrl		
	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.		
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
30	Reserved		
29	CmptCtrl		
	Format:	MBZ	
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations		

brd - Branch Diverging

	supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.									
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.								
28	<p>PredInv</p> <p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table>	Format:	ExecSize							
Format:	ExecSize									

brd - Branch Diverging						
		This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.				
	15:0	<table border="1" style="width: 100%;"> <tr> <td colspan="2">Header</td> </tr> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>	Header		Format:	Header
Header						
Format:	Header					

Break

break - Break				
Source:	Eulsa			
Length Bias:	4			
<p>The break instruction is used to early-out from the inner most loop, or early out from the inner most switch block. When used in a loop, upon execution, the break instruction terminates the loop for all execution channels enabled. If all the enabled channels hit the break instruction, jump to the instruction referenced by JIP. JIP should be the offset to the end of the inner most conditional or loop block, UIP should be the offset to the while instruction of the loop block. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case</p> <p>The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.</p>				
<p>Format:</p> <pre>[(pred)] break (exec_size) JIP UIP</pre>				
Syntax				
<pre>[(pred)] break (exec_size) imm32 imm32</pre>				
Pseudocode				
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { PcIP[n] = IP + UIP; else { PcIP[n] = IP + 1; } } if (PcIP != (IP + 1)) { // all channels Jump(IP + JIP); }</pre>				
Predication	Conditional Modifier	Saturation	Source Modifier	
Y	N	N	N	
DWord	Bit	Description		
0..3	127:96	Reserved		
		Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ		
	127:96	JIP		
		Exists If:	([Src0.IsImm]==true)	
		Format:	S31	
			The byte-aligned jump distance if a jump is taken for the channel.	
	95:80	Reserved		
Exists If:		([Src0.IsImm]==false)		
Format:	MBZ			
95:64	Reserved			
	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)		

break - Break		
	Format:	MBZ
95:64	UIP	
	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)
	Format:	S31
	The byte aligned jump distance if a jump is taken for the instruction.	
79:66	Src0.Operand	
	Exists If:	([Src0.IsImm]==false)
	Format:	DirectOperand
65:64	Reserved	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
63:50	Dst.Operand	
	Format:	DirectOperand
49:48	Reserved	
	Format:	MBZ
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value	
	Value	Name
	0	false
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value	
	Value	Name
	0	false
	1	true
45:34	Reserved	
	Format:	MBZ
33	BranchCtrl	
	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
		Description

break - Break

	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:		MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
	Format:		PredCtrl
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0]		
	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum		
	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the		

break - Break			
	<p>destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Byte Scattered Read MSD

MSDOR_BS - Byte Scattered Read MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: Enable If set, indicates that the message includes the header.
	18	Legacy Message Default Value: 0h

MSDOR_BS - Byte Scattered Read MSD							
	<table border="1"> <tr> <td>Format:</td> <td>Opcode</td> </tr> <tr> <td colspan="2">Legacy Message</td> </tr> </table>	Format:	Opcode	Legacy Message			
Format:	Opcode						
Legacy Message							
17:14	<p>Message Type</p> <table border="1"> <tr> <td>Default Value:</td> <td>04h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> <tr> <td colspan="2">Byte Scattered Read message</td> </tr> </table>	Default Value:	04h	Format:	Opcode	Byte Scattered Read message	
Default Value:	04h						
Format:	Opcode						
Byte Scattered Read message							
13	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
12	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
11:10	<p>Data Elements</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_DS</td> </tr> <tr> <td colspan="2">Specifies the number of Bytes to be read or written per Dword</td> </tr> </table>	Format:	MDC_DS	Specifies the number of Bytes to be read or written per Dword			
Format:	MDC_DS						
Specifies the number of Bytes to be read or written per Dword							
9	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
8	<p>SIMD Mode</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_SM2</td> </tr> <tr> <td colspan="2">Specifies the SIMD mode of the message (number of slots processed)</td> </tr> </table>	Format:	MDC_SM2	Specifies the SIMD mode of the message (number of slots processed)			
Format:	MDC_SM2						
Specifies the SIMD mode of the message (number of slots processed)							
7:0	<p>Binding Table Index</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_BTS_SLM_A32</td> </tr> <tr> <td colspan="2">Specifies the Binding Table Index for the message</td> </tr> </table>	Format:	MDC_BTS_SLM_A32	Specifies the Binding Table Index for the message			
Format:	MDC_BTS_SLM_A32						
Specifies the Binding Table Index for the message							

Byte Scattered Write MSD

MSD0W_BS - Byte Scattered Write MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: Enable If set, indicates that the message includes the header.
	18	Legacy Message Default Value: 0h

MSD0W_BS - Byte Scattered Write MSD

	Format:	Opcode
	Legacy Message	
17:14	Message Type	
	Default Value:	0Ch
	Format:	Opcode
	Byte Scattered Write message	
13:12	Reserved	
	Format:	MBZ
11:10	Data Elements	
	Format:	MDC_DS
	Specifies the number of Bytes to be read or written per Dword	
9	Reserved	
	Format:	MBZ
8	SIMD Mode	
	Format:	MDC_SM2
	Specifies the SIMD mode of the message (number of slots processed)	
7:0	Binding Table Index	
	Format:	MDC_BTS_SLM_A32
	Specifies the Binding Table Index for the message	

Call

call - Call			
Source:	Eulsa		
Length Bias:	4		
<p>The call instruction jumps to a subroutine. It can be predicated or non-predicated. If non-predicated, all enabled channels jump to the subroutine. If predicated, only the channels enabled by PMask jump to the subroutine; the rest of the channels move to the next instruction after the call instruction. If none of the channels jump into the subroutine, the call instruction is treated as a nop. In case of a jump, the call instruction stores the return IP onto the first DWord of the destination register and stores the CallMask in the second DWord of the destination register. When SPF is on, the predication control must be scalar.</p> <p>The following section describes JIP, the jump offset. JIP can be an immediate or register value. When a jump occurs, this value is added to IP pre-increment. In instruction binary, JIP is at location src1 and src0 must be null. The GRF register must be put (for example, by the assembler) at dst location. Format: [(pred)] call (exec_size) dst JIP</p> <p>Format: [(pred)] call (exec_size) dst JIP</p>			
Restriction			
<p>The call instruction must have DWord source and destination type, and the destination must be QWord aligned.</p> <p>A call instruction must target an instruction with Switch set; in addition, the instruction following the call must also have Switch set.</p>			
Syntax			
[(pred)] call (exec_size) reg imm32 [(pred)] call (exec_size) reg reg32			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { PcIP[n] = IP + JIP; CallMask[n] = 1; } else { PcIP[n] = IP + 1; CallMask[n] = 0; } if (PcIP[n] != (IP + 1)) { // any channel jumped dst.chan[0] = IP + 1; dst.chan[1] = CallMask; Jump(IP + JIP); }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
	Format:	MBZ	
	127:96	JIP	

call - Call		
	Exists If:	([Src0.IsImm]==true)
	Format:	S31
The byte-aligned jump distance if a jump is taken for the channel		
95:80	Reserved	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
95:64	Reserved	
	Exists If:	([Src0.IsImm]==true)
	Format:	MBZ
79:66	Src0.Operand	
	Exists If:	([Src0.IsImm]==false)
	Format:	DirectOperand
65:64	Reserved	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
63:50	Dst.Operand	
	Format:	DirectOperand
49:47	Reserved	
	Format:	MBZ
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false
	1	true
45:34	Reserved	
	Format:	MBZ
33	BranchCtrl	
	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
		Description

call - Call

	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:		MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
	Format:		PredCtrl
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0]		
	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum		
	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the		

call - Call			
	<p>destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Call Absolute

calla - Call Absolute			
Source:	Eulsa		
Length Bias:	4		
<p>The calla instruction jumps to a subroutine. It can be predicated or non-predicated. If non-predicated, all enabled channels jump to the subroutine. If predicated, only the channels enabled by PMask jump to the subroutine; the rest of the channels move to the next instruction after the calla instruction. If none of the channels jump into the subroutine, the calla instruction is treated as a nop. In case of a jump, the call instruction stores the return IP onto the first DWord of the destination register and stores the CallMask in the second DWord of the destination register. If SPF is ON, none of the PcIP are updated. When SPF is on, the predication control must be scalar. The difference between calla and call is that calla uses JIP as the IP value rather than adding it to the IP value.</p>			
<p>Format:</p> <pre>[(pred)] calla (exec_size) dst JIP</pre>			
Restriction			
<p>The calla instruction must have DWord source and destination type, and the destination must be QWord-aligned.</p>			
Syntax			
<pre>[(pred)] calla (exec_size) reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { PcIP[n] = JIP; CallMask[n] = 1; } else { PcIP[n] = IP + 1; CallMask[n] = 0; } } if (PcIP[n] != (IP + 1)) { // any channel jumped dst.chan[0] = IP + 1; dst.chan[1] = CallMask; Jump(JIP); }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
		Format:	MBZ
	127:96	JIP	
		Exists If:	([Src0.IsImm]==true)
		Format:	S31
		The byte-aligned jump distance if a jump is taken for the channel	
95:80	Reserved		

calla - Call Absolute

	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
95:64	Reserved		
	Exists If:	([Src0.IsImm]==true)	
	Format:	MBZ	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false)	
	Format:	DirectOperand	
65:64	Reserved		
	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
63:50	Dst.Operand		
	Format:	DirectOperand	
49:47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
	1	true	
45:34	Reserved		
	Format:	MBZ	
33	BranchCtrl		
	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.		
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		

calla - Call Absolute

		Format:	MBZ
		<p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p>	
		Value	Name
		Description	
		0	NoCompaction [Default]
		1	Compacted
		<p>No compaction. 128-bit native instruction supporting all instruction options.</p> <p>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</p>	
28	<p>PredInv</p> <p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p>		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	<p>PredCtrl</p>		
	Format:	PredCtrl	
	<p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p>		
	Format:	ChanOff	

calla - Call Absolute			
	This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Compare

cmp - Compare				
Source:	Eulsa			
Length Bias:	4			
<p>The cmp instruction performs component-wise comparison of src0 and src1 and stores the results in the selected flag register and in dst. It takes component-wise subtraction of src0 and src1, evaluating the conditional code (excluding NS signal) based on the conditional modifier, and storing the conditional bits in bit-packed form in the destination flag register and all bits of dst channels. If the dst is not null, for the enabled channels, then all bits of the destination channel will contain the flag value for the channel. When the instruction operates on packed word format, one general register may store up to 16 such comparison results. In DWord format, one general register may store up to 8 results. A conditional modifier must be specified; the conditional modifier field cannot be 0000b. The comparison does not use the NS (NaN source) signals, as described in the Creating Conditional Flags section. Accordingly the conditional modifier should not be .u (unordered). For each enabled channel 0b or 1b is assigned to the appropriate flag bit and 0/all zeros or all ones (e.g, byte 0xFF, word 0xFFFF, DWord 0xFFFFFFFF) is assigned to dst. When any source type is floating-point, the cmp instruction obeys the rules described in the tables in the Floating Point Modes section of the Data Types chapter.</p>				
Format:	[(pred)] cmp[.cm] (exec_size) dst src0 src1			
Syntax				
[(pred)] cmp[.cm] (exec_size) reg reg reg		[(pred)] cmp[.cm] (exec_size) reg reg imm32		
Pseudocode				
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { results[n] = src0.chan[n] - src1.chan[n]; bitMask[n] = Condition(results[n]); dst.chan[n] = bitMask[n]; // All bits for dst channel flag#.bit[n]= bitMask[n]; } } </pre>				
Predication	Conditional Modifier	Saturation	Source Modifier	
Y	Y	N	Y	
Src Types			Dst Types	
*B,*W,*D			*B,*W,*D	
*B,*W,*D			F	
F			F	
HF			HF	

cmp - Compare		
*B,*W,*D		HF
HF, F		HF, F
DWord	Bit	Description
0..3	127:126	Reserved
		Exists If: ([Src1.IsImm]==false)
		Format: MBZ
	127:96	Src1.ImmValue[31:0]
		Exists If: ([Src1.IsImm]==true)
	125:122	Reserved
		Exists If: ([Src1.IsImm]==false)
		Format: MBZ
	121:120	Src1.Mod
		Exists If: ([Src1.IsImm]==false)
		Format: SrcMod
	119:116	Src1.VertStride
		Exists If: ([Src1.IsImm]==false)
		Format: VertStride
	115:113	Src1.Width
		Exists If: ([Src1.IsImm]==false)
Format: Width		
112	Src1.AddrMode	
	Exists If: ([Src1.IsImm]==false)	
	Format: AddrMode	
111:98	Src1.Operand	
	Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)	
	Format: DirectOperand	
111:98	Src1.Operand	
	Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)	
	Format: IndirectOperand	
97:96	Src1.HorzStride	
	Exists If: ([Src1.IsImm]==false)	
	Format: HorzStride	
95:92	CondCtrl	
	Format: FlagModifier	
91:88	Src1.DataType	
	Exists If: ([Src1.IsImm]==false)	

cmp - Compare

	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true

cmp - Compare										
45:44	Src0.Mod	Format: SrcMod								
43:40	Src0.DataType	Exists If: ([Src0.IsImm]==false) Format: RegDataType								
43:40	Src0.DataType	Exists If: ([Src0.IsImm]==true) Format: ImmDataType								
39:36	Dst.DataType	Format: RegDataType								
35	Dst.AddrMode	Format: AddrMode								
34	Saturate	Format: Saturate								
33	AccWrCtrl	Format: AccWrCtrl								
32	AtomicCtrl	Format: AtomicCtrl								
31	MaskCtrl	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".								
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td>1</td> <td>NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description								
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.								
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.								
30	Reserved									
29	CmptCtrl	Format: MBZ								
		Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.								
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.			
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								

cmp - Compare

	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	<p>PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>			Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>			Format:	PredCtrl							
Format:	PredCtrl											
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>											
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>											
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>			Format:	ChanOff							
Format:	ChanOff											
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>			Format:	ExecSize							
Format:	ExecSize											
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>			Format:	Header							
Format:	Header											

Compare NaN

cmpn - Compare NaN			
Source:	Eulsa		
Length Bias:	4		
<p>The cmpn instruction performs component-wise special-NaN comparison of src0 and src1 and stores the results in the selected flag register and in dst. It takes component-wise subtraction of src0 and src1, evaluating the conditional signals including NS based on the conditional modifier, and storing the conditional flag bits in bit-packed form in the destination flag register and all bits of dst channels. If the dst is not null, for the enabled channels, then all bits of the destination channel will contain the flag value for the channel. When the instruction operates on packed word format, one general register may store up to 16 such comparison results. In DWord format, one general register may store up to 8 results. A conditional modifier must be specified; the conditional modifier field cannot be 0000b. More information about the conditional signals used is in the Creating Conditional Flags section. For each enabled channel 0b or 1b is assigned to the appropriate flag bit and 0/all zeros or all ones (e.g, byte 0xFF, word 0xFFFF, DWord 0xFFFFFFFF) is assigned to dst. Min/Max instructions use cmpn to select the destination from the input sources (see the Min Max of Floating Point Numbers section for details).</p>			
Format:	<code>[(pred)] cmpn[.cmod] (exec_size) dst src0 src1</code>		
Restriction			
.l and .ge are the only two conditional modifiers are supported for this instruction.			
Syntax			
<code>[(pred)] cmpn[.cmod] (exec_size) reg reg reg [(pred)] cmpn[.cmod] (exec_size) reg reg imm32</code>			
Pseudocode			
<pre> Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { results[n] = src0.chan[n] - src1.chan[n]; bitMask[n] = ConditionNaN(results[n]); dst.chan[n][0] = bitMask[n]; // All bits for dst channel flag#.bit[n] = bitMask[n]; } } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
*B,*W,*D	F		

cmpn - Compare NaN

DWord	Bit	Description				
F	F					
HF	HF					
0..3	127:126	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)	Format:	MBZ
	Exists If:	([Src1.IsImm]==false)				
	Format:	MBZ				
	127:96	Src1.ImmValue[31:0] <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==true)</td> </tr> </table>	Exists If:	([Src1.IsImm]==true)		
	Exists If:	([Src1.IsImm]==true)				
	125:122	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)	Format:	MBZ
	Exists If:	([Src1.IsImm]==false)				
	Format:	MBZ				
	121:120	Src1.Mod <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>SrcMod</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)	Format:	SrcMod
	Exists If:	([Src1.IsImm]==false)				
	Format:	SrcMod				
	119:116	Src1.VertStride <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>VertStride</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)	Format:	VertStride
	Exists If:	([Src1.IsImm]==false)				
	Format:	VertStride				
	115:113	Src1.Width <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>Width</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)	Format:	Width
Exists If:	([Src1.IsImm]==false)					
Format:	Width					
112	Src1.AddrMode <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>AddrMode</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)	Format:	AddrMode	
Exists If:	([Src1.IsImm]==false)					
Format:	AddrMode					
111:98	Src1.Operand <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)</td> </tr> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)	Format:	DirectOperand	
Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)					
Format:	DirectOperand					
111:98	Src1.Operand <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)</td> </tr> <tr> <td>Format:</td> <td>IndirectOperand</td> </tr> </table>	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)	Format:	IndirectOperand	
Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)					
Format:	IndirectOperand					
97:96	Src1.HorzStride <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>HorzStride</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)	Format:	HorzStride	
Exists If:	([Src1.IsImm]==false)					
Format:	HorzStride					
95:92	CondCtrl <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Format:</td> <td>FlagModifier</td> </tr> </table>	Format:	FlagModifier			
Format:	FlagModifier					
91:88	Src1.DataType <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src1.IsImm]==false)</td> </tr> </table>	Exists If:	([Src1.IsImm]==false)			
Exists If:	([Src1.IsImm]==false)					

cmpn - Compare NaN

	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true

cmpn - Compare NaN

		cmpn - Compare NaN																																						
<div style="display: flex; flex-direction: column; gap: 10px;"> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 45:44 Src0.Mod </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">SrcMod</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 43:40 Src0.DataType </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td style="text-align: right;">RegDataType</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 43:40 Src0.DataType </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([Src0.IsImm]==true)</td> </tr> <tr> <td>Format:</td> <td style="text-align: right;">ImmDataType</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 39:36 Dst.DataType </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">RegDataType</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 35 Dst.AddrMode </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">AddrMode</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 34 Saturate </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">Saturate</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 33 AccWrCtrl </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">AccWrCtrl</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 32 AtomicCtrl </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">AtomicCtrl</td> </tr> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 31 MaskCtrl </div> <div style="border: 1px solid black; padding: 2px;"> <p>Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: left;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td>1</td> <td>NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table> </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 30 Reserved </div> </div> <div style="border: 1px solid black; padding: 2px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid black;"> 29 CmptCtrl </div> <div style="border: 1px solid black; padding: 2px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">MBZ</td> </tr> </table> <p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: left;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> </tbody> </table> </div> </div> </div>	Format:	SrcMod	Exists If:	([Src0.IsImm]==false)	Format:	RegDataType	Exists If:	([Src0.IsImm]==true)	Format:	ImmDataType	Format:	RegDataType	Format:	AddrMode	Format:	Saturate	Format:	AccWrCtrl	Format:	AtomicCtrl	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	Format:	MBZ	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.			
	Format:	SrcMod																																						
	Exists If:	([Src0.IsImm]==false)																																						
	Format:	RegDataType																																						
	Exists If:	([Src0.IsImm]==true)																																						
	Format:	ImmDataType																																						
	Format:	RegDataType																																						
	Format:	AddrMode																																						
	Format:	Saturate																																						
	Format:	AccWrCtrl																																						
	Format:	AtomicCtrl																																						
	Value	Name	Description																																					
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.																																					
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.																																					
Format:	MBZ																																							
Value	Name	Description																																						
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.																																						

cmpn - Compare NaN

	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	<p>PredInv</p> <p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>			Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>			Format:	PredCtrl							
Format:	PredCtrl											
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>											
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>											
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>			Format:	ChanOff							
Format:	ChanOff											
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>			Format:	ExecSize							
Format:	ExecSize											
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>			Format:	Header							
Format:	Header											

Conditional Select

csel - Conditional Select			
Source:	Eulsa		
Length Bias:	4		
<p>The csel instruction selectively moves components in src0 or src1 to the dst based on the result of the compare of src2 with zero. If the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst. The csel instruction provides the function of a cmp followed by sel. The instruction must not be used if cmpn is required. The instruction does not update the flag register. The comparison follows the same rule as cmp instruction for that data type.</p> <p>When Access Mode is Align1, accumulator may be used as source or destination.</p>			
Format:	<pre>csel (exec_size) dst src0 src1 src2</pre>		
Syntax			
csel[.cmod] (exec_size) reg reg reg reg			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { bitMask[n] = 0; if (EMask.chan[n]) { result[n] = src2.chan[n] - 0; bitMask[n] = Condition(result[n]); if (bitMask[n] = 1) { dst.chan[n] = src0.chan[n]; } else { dst.chan[n] = src1.chan[n]; } } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
N	Y	Y	Y
Src Types	Dst Types		
F	F		
HF	HF		
D	D		
W	W		
DWord	Bit	Description	
0..3	127:114	Src2.Operand	
		Exists If:	((Src2.IsImm) == false) AND ((Header)[Opcode] != madm)
	Format:	DirectOperand	
127:114	Src2.Operand		
	Exists If:	((Src2.IsImm) == false) AND ((Header)[Opcode] == madm)	
Format:	MacroOperand		
127:112	Src2.ImmValue[15:0]		
Exists If:	((Src2.IsImm) == true)		

csel - Conditional Select

113:112	Src2.HorzStride	
	Exists If:	([Src2.IsImm]==false)
	Format:	HorzStride
111:98	Src1.Operand	
	Exists If:	([Header][Opcode]!=madm)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	([Header][Opcode]==madm)
	Format:	MacroOperand
97:96	Src1.HorzStride	
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91	Src1.VertStride[1]	
	Format:	TernaryVertStride[1]
90:88	Src1.DataType	
	Format:	TernaryDataType
87:86	Src1.Mod	
	Format:	SrcMod
85:84	Src2.Mod	
	Format:	SrcMod
83	Src1.VertStride[0]	
	Format:	TernaryVertStride[0]
82:80	Src2.DataType	
	Format:	TernaryDataType
79:66	Src0.Operand	
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]!=madm)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]==madm)
	Format:	MacroOperand
79:64	Src0.ImmValue[15:0]	
	Exists If:	([Src0.IsImm]==true)
65:64	Src0.HorzStride	
	Exists If:	([Src0.IsImm]==false)
	Format:	HorzStride

csel - Conditional Select

63:50	Dst.Operand		
		Exists If:	([Header][Opcode]!=madm)
		Format:	DirectOperand
Programming Notes			
The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,			
63:50	Dst.Operand		
		Exists If:	([Header][Opcode]==madm)
		Format:	MacroOperand
49	Reserved		
		Format:	MBZ
48	Dst.HorzStride	This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand.	
		Value	Name
		0	1 element
		1	2 element
47	Src2.IsImm	This field indicate that Source 2 operand is carrying an immediate value.	
		Value	Name
		0	false
		1	true
46	Src0.IsImm	This field indicate that Source 0 operand is carrying an immediate value.	
		Value	Name
		0	false
		1	true
45:44	Src0.Mod		
		Format:	SrcMod
43	Src0.VertStride[1]		
		Format:	TernaryVertStride[1]
42:40	Src0.DataType		
		Format:	TernaryDataType
39	ExecDataType	This field indicate the datatype mode of ternary instruction. Integer or Float.	
		Value	Name
		0	Integer

csel - Conditional Select

	1	Float									
38:36	Dst.DataType Format: TernaryDataType										
35	Src0.VertStride[0] Format: TernaryVertStride[0]										
34	Saturate Format: Saturate										
33	AccWrCtrl Format: AccWrCtrl										
32	AtomicCtrl Format: AtomicCtrl										
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0". <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>		Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.									
30	Reserved										
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>		Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields.										

csel - Conditional Select			
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
	Format:		PredCtrl
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff		
	Format:		ChanOff
	This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.		
18:16	ExecSize		
	Format:		ExecSize
	This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.		
15:0	Header		
	Format:		Header

Constant Cache Dword Scattered Read MSD

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: Enable If set, indicates that the message includes the header.
	18	Legacy Message Default Value: 0h

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD

	Format:	Opcode
	Legacy Message	
17:14	Message Type	
	Default Value:	03h
	Format:	Opcode
	Dword Scattered Read message	
13	Invalidate After Read	
	Format:	MDC_IAR
	Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs	
12:10	Reserved	
	Format:	MBZ
9	Legacy SIMD Mode	
	Default Value:	1h
	Format:	Opcode
	Must be set for compatibility.	
8	SIMD Mode	
	Format:	MDC_SM2
	Specifies the SIMD mode of the message (number of slots processed)	
7:0	Binding Table Index	
	Format:	MDC_BTS
	Specifies the Binding Table Index for the message	

Constant Cache Oword Aligned Block Read MSD

MSD_CC_OWAB - Constant Cache Oword Aligned Block Read MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18	Reserved Format: _____ MBZ
	17:14	Message Type Default Value: _____ 01h Format: _____ Opcode Oword Aligned Block Read Constant Cache message
	13:11	Reserved Format: _____ MBZ
	10:8	Data Elements Format: _____ MDC_DB_OW Specifies the number of contiguous Owords to be read
	7:0	Binding Table Index Format: _____ MDC_BTS Specifies the Binding Table Index for the message

Constant Cache Oword Block Read MSD

MSD_CC_OWB - Constant Cache Oword Block Read MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18	Reserved Format: _____ MBZ
	17:14	Message Type Default Value: _____ 00h Format: _____ Opcode Oword Block Read Constant Cache message
	13	Invalidate After Read Format: _____ MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	Reserved Format: _____ MBZ
	10:8	Data Elements Format: _____ MDC_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	Binding Table Index Format: _____ MDC_BTS Specifies the Binding Table Index for the message

Continue

cont - Continue			
Source:	Eulsa		
Length Bias:	4		
<p>The cont instruction disables execution for the subset of channels for the remainder of the current loop iteration. Channels remain disabled until right before the while instruction or right before the condition check code block for the while instruction. If all enabled channels hit this instruction, jump to the instruction referenced by JIP where execution continues. UIP should always reference the loop's associated while instruction. JIP should point to the last instruction of the inner most conditional block if the cont instruction is inside a conditional block. In case of the break instruction directly under the loop, the JIP and the UIP are the same. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.</p> <p>The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.</p>			
Format:	<pre>[(pred)] cont (exec_size) JIP UIP</pre>		
Restriction			
The execution size must be the same for the while, break, and cont instructions of the same code block.			
Syntax			
<pre>[(pred)] cont (exec_size) imm32 imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.channel[n]) { if (PMask[n]) { // PMask is for all channels enabled for the cont instruction. PcIP[n] = IP + UIP; } } else { PcIP[n] = IP + 1; } } } for (n = exec_size; n < 32; n++) { PcIP[n] = IP + 1; } } if (PcIP != (IP + 1)) { // all channels true Jump(IP + JIP); }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
		Format:	MBZ
	127:96	JIP	
		Exists If:	([Src0.IsImm]==true)
		Format:	S31
		The byte-aligned jump distance if a jump is taken for the channel.	

cont - Continue

cont - Continue							
95:80	<p>Reserved</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	MBZ		
Exists If:	([Src0.IsImm]==false)						
Format:	MBZ						
95:64	<p>Reserved</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==true) AND ([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)	Format:	MBZ		
Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)						
Format:	MBZ						
95:64	<p>UIP</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==true) AND ([Src1.IsImm]==true)</td> </tr> <tr> <td>Format:</td> <td>S31</td> </tr> </table> <p>The byte aligned jump distance if a jump is taken for the instruction.</p>	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)	Format:	S31		
Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)						
Format:	S31						
79:66	<p>Src0.Operand</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	DirectOperand		
Exists If:	([Src0.IsImm]==false)						
Format:	DirectOperand						
65:64	<p>Reserved</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	MBZ		
Exists If:	([Src0.IsImm]==false)						
Format:	MBZ						
63:50	<p>Dst.Operand</p> <table border="1"> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Format:	DirectOperand				
Format:	DirectOperand						
49:48	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
47	<p>Src1.IsImm</p> <p>This field indicate that Source 1 operand is carrying an immediate value</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>false</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </tbody> </table>	Value	Name	0	false	1	true
Value	Name						
0	false						
1	true						
46	<p>Src0.IsImm</p> <p>This field indicate that Source 0 operand is carrying an immediate value</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>false</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </tbody> </table>	Value	Name	0	false	1	true
Value	Name						
0	false						
1	true						
45:34	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
33	<p>BranchCtrl</p> <p>This field is used by <i>goto</i>, <i>if</i>, and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.</p>						

cont - Continue

32	AtomicCtrl		
		Format:	AtomicCtrl
31	MaskCtrl	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
		Value	Name
		Description	
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
		Format:	MBZ
Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.			
		Value	Name
		Description	
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
		Value	Name
		Description	
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
		Format:	PredCtrl
This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.			

cont - Continue

23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Count Bits Set

cbit - Count Bits Set			
Source:	Eulsa		
Length Bias:	4		
The cbit instruction counts component-wise the total bits set in src0 and stores the resulting counts in dst.			
Format:	<code>[(pred)] cbit (exec_size) dst src0</code>		
Restriction			
No accumulator access, implicit or explicit.			
Syntax			
<code>[(pred)] cbit (exec_size) reg reg</code> <code>[(pred)] cbit (exec_size) reg imm32</code>			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) while (UD cnt = 0; UD val = src0.chan[n]; val) { if (val & 1) { cnt ++; } dst.chan[n] = cnt; } val = val >> 1; } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
Src Types	Dst Types		
UB, UW, UD	UD		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0] Exists If: <code>([Src0.IsImm]==true)</code>	
	95:92	CondCtrl Exists If: <code>([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))</code> Format: FlagModifier	
	95:64	Src0.ImmValue[63:32] Exists If: <code>([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))</code>	
	87:84	Src0.VertStride Exists If: <code>([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))</code> Format: VertStride	
	83:81	Src0.Width	

cbit - Count Bits Set

	Exists If:	([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))	
	Format:	Width	
80	Src0.AddrMode		
	Exists If:	([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))	
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))) AND ([Src0.AddrMode] == Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))) AND ([Src0.AddrMode] == Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode] == Indirect)	
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode] == Direct)	
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		

cbit - Count Bits Set

		Exists If:	([Src0.IsImm]==false)
		Format:	RegDataType
43:40	Src0.DataType		
		Exists If:	([Src0.IsImm]==true)
		Format:	ImmDataType
39:36	Dst.DataType		
		Format:	RegDataType
35	Dst.AddrMode		
		Format:	AddrMode
34	Saturate		
		Format:	Saturate
33	AccWrCtrl		
		Format:	AccWrCtrl
32	AtomicCtrl		
		Format:	AtomicCtrl
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
		Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask		

cbit - Count Bits Set

	<p>for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize							
Format:	ExecSize									
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>	Format:	Header							
Format:	Header									

Dot Product 4 Accumulate

dp4a - Dot Product 4 Accumulate			
Source:	Eulsa		
Length Bias:	4		
<p>DP4A is a packed four-wide integer dot product and accumulate operation. Each source's 32-bit channel value is treated as four element vector of 8-bit integer values. The operation performs a 32-bit precision dot product of those four bytes and adds it with a 32-bit accumulator (typically a GRF, not necessarily an acc# reg).</p>			
Format:	[(pred)] dp4a (exec_size) dst src0 src1 src2		
Programming Notes			
<pre>EXAMPLE (SIMD1 for simplicity): mov (1) r1.0:d 0x0102037F:d // (char4) (0x1,0x2,0x3,0x7F) mov (1) r2.0:d 50:d dp4a (1) r3.0:d r2:d r1:d r1:d // r3.0 = 50 + (0x1*0x1 + 0x2*0x2 + 0x3*0x3 + 0x7F*0x7F) // = 50 + (1 + 4 + 9 + 16129) // = 16193</pre>			
Restriction			
All three-source instructions have certain restrictions, described in Instruction Formats.			
Only one of src0 or src1 operand may be an the accumulator register (acc#).			
Syntax			
<pre>[(pred)] dp4a (exec_size) reg reg reg reg [(pred)] dp4a (exec_size) reg reg reg imm16 [(pred)] dp4a (exec_size) reg imm16 reg reg</pre>			
Pseudocode			
<pre>Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst[n] = src0.chan[n] + src1.chan[n][7:0] * src2.chan[n][7:0] + src1.chan[n][15:8] * src2.chan[n][15:8] + src1.chan[n][23:16] * src2.chan[n][23:16] + src1.chan[n][31:24] * src2.chan[n][31:24]; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	N
Src Types	Dst Types		

dp4a - Dot Product 4 Accumulate

*D	*D			
DWord	Bit	Description		
0..3	127:114	Src2.Operand		
		Exists If:	((Src2.IsImm) == false) AND ([Header][Opcode] != madm)	
		Format:	DirectOperand	
	127:114	Src2.Operand		
		Exists If:	((Src2.IsImm) == false) AND ([Header][Opcode] == madm)	
		Format:	MacroOperand	
	127:112	Src2.ImmValue[15:0]		
		Exists If:	((Src2.IsImm) == true)	
	113:112	Src2.HorzStride		
		Exists If:	((Src2.IsImm) == false)	
		Format:	HorzStride	
	111:98	Src1.Operand		
		Exists If:	([Header][Opcode] != madm)	
		Format:	DirectOperand	
	111:98	Src1.Operand		
		Exists If:	([Header][Opcode] == madm)	
		Format:	MacroOperand	
	97:96	Src1.HorzStride		
Format:		HorzStride		
95:92	CondCtrl			
	Format:	FlagModifier		
91	Src1.VertStride[1]			
	Format:	TernaryVertStride[1]		
90:88	Src1.DataType			
	Format:	TernaryDataType		
87:86	Src1.Mod			
	Format:	SrcMod		
85:84	Src2.Mod			
	Format:	SrcMod		
83	Src1.VertStride[0]			
	Format:	TernaryVertStride[0]		
82:80	Src2.DataType			
	Format:	TernaryDataType		
79:66	Src0.Operand			

dp4a - Dot Product 4 Accumulate

	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]!=madm)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]==madm)	
	Format:	MacroOperand	
79:64	Src0.ImmValue[15:0]		
	Exists If:	([Src0.IsImm]==true)	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm]==false)	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Header][Opcode]!=madm)	
	Format:	DirectOperand	
Programming Notes			
The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,			
63:50	Dst.Operand		
	Exists If:	([Header][Opcode]==madm)	
	Format:	MacroOperand	
49	Reserved		
	Format:	MBZ	
48	Dst.HorzStride		
	This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand.		
	Value	Name	
	0	1 element	
	1	2 element	
47	Src2.IsImm		
	This field indicate that Source 2 operand is carrying an immediate value.		
	Value	Name	
	0	false	
	1	true	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
	1	true	

dp4a - Dot Product 4 Accumulate

45:44	Src0.Mod	
	Format:	SrcMod
43	Src0.VertStride[1]	
	Format:	TernaryVertStride[1]
42:40	Src0.DataType	
	Format:	TernaryDataType
39	ExecDataType	
	This field indicate the datatype mode of ternary instruction. Integer or Float.	
	Value	Name
	0	Integer
	1	Float
38:36	Dst.DataType	
	Format:	TernaryDataType
35	Src0.VertStride[0]	
	Format:	TernaryVertStride[0]
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
	0	Normal [Default]
	1	NoMask
		Description
		Normal. Per channel write enable used for final write enable generation.
		NoMask.Skips the check for PclP[n] == ExlP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	

dp4a - Dot Product 4 Accumulate

		Value	Name	Description		
		0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.		
		1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.		
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields					
		0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.		
		1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.		
27:24	PredCtrl Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">PredCtrl</td></tr></table> This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.					PredCtrl
	PredCtrl					
23	FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.					
22	FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.					
21:19	ChanOff Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">ChanOff</td></tr></table> This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.					ChanOff
	ChanOff					
18:16	ExecSize Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">ExecSize</td></tr></table> This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.					ExecSize
	ExecSize					
15:0	Header Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">Header</td></tr></table>					Header
	Header					

Dword Atomic Counter with Return Data Operation MSD

MSD1R_DWAC - Dword Atomic Counter with Return Data Operation MSD										
Source:		EuSubFunctionDataPort1								
Length Bias:		1								
DWord	Bit	Description								
0	31	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
	30	Packed Data Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF).</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th colspan="2" style="text-align: center;">Restriction</th> </tr> <tr> <td colspan="2" style="text-align: center;">Only 32-bit data packing is supported at this time.</td> </tr> </table>	Default Value:	0 32 bit	Format:	Enable	Restriction		Only 32-bit data packing is supported at this time.	
	Default Value:	0 32 bit								
	Format:	Enable								
	Restriction									
	Only 32-bit data packing is supported at this time.									
	29	Packed Address Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.</p>	Default Value:	0 32 bit	Format:	Enable				
Default Value:	0 32 bit									
Format:	Enable									
28:25	Message Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4							
Format:	U4									
24:20	Response Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5							
Format:	U5									
19	Header Present <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="color: red;">MDC_MHR</td> </tr> </table> <p>Indicates that the message requires a header</p>	Format:	MDC_MHR							
Format:	MDC_MHR									
18:14	Message Type									

MSD1R_DWAC - Dword Atomic Counter with Return Data Operation MSD

		Default Value:	0Bh
		Format:	Opcode
	Atomic Counter Operation message		
13	Return Data Control		
		Default Value:	1h
		Format:	Opcode
	Specifies that return data is sent back to the thread.		
12	SIMD Mode		
	Format:	MDC_SM2RS	
	Specifies the SIMD mode of the message (number of slots processed)		
11:8	Atomic Integer Operation		
	Format:	MDC_AOP	
	Specifies the atomic integer operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS	
	Specifies the Binding Table Index for the message		

Dword Atomic Counter Write Only Operation MSD

MSD1W_DWAC - Dword Atomic Counter Write Only Operation MSD								
Source:		EuSubFunctionDataPort1						
Length Bias:		1						
DWord	Bit	Description						
0	31	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	30	Packed Data Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF).</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Restriction</th> </tr> <tr> <td>Only 32-bit data packing is supported at this time.</td> </tr> </table>	Default Value:	0 32 bit	Format:	Enable	Restriction	Only 32-bit data packing is supported at this time.
	Default Value:	0 32 bit						
	Format:	Enable						
	Restriction							
	Only 32-bit data packing is supported at this time.							
	29	Packed Address Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.</p>	Default Value:	0 32 bit	Format:	Enable		
	Default Value:	0 32 bit						
	Format:	Enable						
28:25	Message Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4					
Format:	U4							
24:20	Response Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5					
Format:	U5							
19	Header Present <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="color: red;">MDC_MHR</td> </tr> </table> <p>Indicates that the message requires a header</p>	Format:	MDC_MHR					
Format:	MDC_MHR							
18:14	Message Type							

MSD1W_DWAC - Dword Atomic Counter Write Only Operation MSD

		Default Value:	0Bh
		Format:	Opcode
	Atomic Counter Operation message		
	13	Return Data Control	
		Default Value:	0h
		Format:	Opcode
	Specifies that no return data is sent back to the thread.		
	12	SIMD Mode	
		Format:	MDC_SM2RS
	Specifies the SIMD mode of the message (number of slots processed)		
	11:8	Atomic Integer Operation	
		Format:	MDC_AOP
	Specifies the atomic integer operation to be performed.		
	7:0	Binding Table Index	
		Format:	MDC_BTS
	Specifies the Binding Table Index for the message		

Dword Scattered Read MSD

MSD0R_DWS - Dword Scattered Read MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: Enable If set, indicates that the message includes the header.
	18	Legacy Message Default Value: 0h

MSD0R_DWS - Dword Scattered Read MSD		
	Format:	Opcode
	Legacy Message	
17:14	Message Type	
	Default Value:	03h
	Format:	Opcode
	Dword Scattered Read message	
13	Invalidate After Read	
	Format:	MDC_IAR
	Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs	
12:10	Reserved	
	Format:	MBZ
9	Legacy SIMD Mode	
	Default Value:	1h
	Format:	Opcode
	Must be set for compatibility.	
8	SIMD Mode	
	Format:	MDC_SM2
	Specifies the SIMD mode of the message (number of slots processed)	
7:0	Binding Table Index	
	Format:	MDC_BTS_A32
	Specifies the Binding Table Index for the message	

Dword Scattered Write MSD

MSD0W_DWS - Dword Scattered Write MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: Enable If set, indicates that the message includes the header.
	18	Legacy Message Default Value: 0h

MSD0W_DWS - Dword Scattered Write MSD		
	Format:	Opcode
	Legacy Message	
17:14	Message Type	
	Default Value:	0Bh
	Format:	Opcode
	Dword Scattered Write message	
13:10	Reserved	
	Format:	MBZ
9	Legacy SIMD Mode	
	Default Value:	1h
	Format:	Opcode
	Must be set for compatibility.	
8	SIMD Mode	
	Format:	MDC_SM2
	Specifies the SIMD mode of the message (number of slots processed)	
7:0	Binding Table Index	
	Format:	MDC_BTS_A32
	Specifies the Binding Table Index for the message	

Dword Typed Atomic Integer with Return Data Operation MSD

MSD1R_DWTAI - Dword Typed Atomic Integer with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type

MSD1R_DWTAI - Dword Typed Atomic Integer with Return Data Operation MSD

		Default Value:	06h
		Format:	Opcode
	Typed Atomic Integer Operation message		
	13	Return Data Control	
		Default Value:	1h
		Format:	Opcode
	Specifies that return data is sent back to the thread.		
	12	Slot Group	
		Format:	MDC_SG2
	Specifies the Slot Group mode of the message (which slots are processed)		
	11:8	Atomic Integer Operation	
		Format:	MDC_AOP
	Specifies the atomic integer operation to be performed.		
	7:0	Binding Table Index	
		Format:	MDC_BTS
	Specifies the Binding Table Index for the message		

Dword Typed Atomic Integer Write Only Operation MSD

MSD1W_DWTAI - Dword Typed Atomic Integer Write Only Operation MSD								
Source:		EuSubFunctionDataPort1						
Length Bias:		1						
DWord	Bit	Description						
0	31	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	30	Packed Data Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF).</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Restriction</th> </tr> <tr> <td>Only 32-bit data packing is supported at this time.</td> </tr> </table>	Default Value:	0 32 bit	Format:	Enable	Restriction	Only 32-bit data packing is supported at this time.
	Default Value:	0 32 bit						
	Format:	Enable						
	Restriction							
	Only 32-bit data packing is supported at this time.							
	29	Packed Address Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.</p>	Default Value:	0 32 bit	Format:	Enable		
Default Value:	0 32 bit							
Format:	Enable							
28:25	Message Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4					
Format:	U4							
24:20	Response Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5					
Format:	U5							
19	Header Present <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="color: red;">MDC_MHP</td> </tr> </table> <p>If set, indicates that the message includes the header.</p>	Format:	MDC_MHP					
Format:	MDC_MHP							
18:14	Message Type							

MSD1W_DWTAI - Dword Typed Atomic Integer Write Only Operation MSD

		Default Value:	06h
		Format:	Opcode
	Typed Atomic Integer Operation message		
13	Return Data Control		
		Default Value:	0h
		Format:	Opcode
	Specifies that no return data is sent back to the thread.		
12	Slot Group		
	Format:	MDC_SG2	
	Specifies the Slot Group mode of the message (which slots are processed)		
11:8	Atomic Integer Operation		
	Format:	MDC_AOP	
	Specifies the atomic integer operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS	
	Specifies the Binding Table Index for the message		

Dword Untyped Atomic Float with Return Data Operation MSD

MSD1R_DWAF - Dword Untyped Atomic Float with Return Data Operation MSD										
Source:		EuSubFunctionDataPort1								
Length Bias:		1								
DWord	Bit	Description								
0	31	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
	30	Packed Data Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF).</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th colspan="2" style="text-align: center;">Restriction</th> </tr> <tr> <td colspan="2" style="text-align: center;">Only 32-bit data packing is supported at this time.</td> </tr> </table>	Default Value:	0 32 bit	Format:	Enable	Restriction		Only 32-bit data packing is supported at this time.	
	Default Value:	0 32 bit								
	Format:	Enable								
	Restriction									
	Only 32-bit data packing is supported at this time.									
	29	Packed Address Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.</p>	Default Value:	0 32 bit	Format:	Enable				
	Default Value:	0 32 bit								
	Format:	Enable								
28:25	Message Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4							
Format:	U4									
24:20	Response Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5							
Format:	U5									
19	Header Present <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="color: red;">MDC_MHP</td> </tr> </table> <p>If set, indicates that the message includes the header.</p>	Format:	MDC_MHP							
Format:	MDC_MHP									
18:14	Message Type									

MSD1R_DWAF - Dword Untyped Atomic Float with Return Data Operation MSD

		Default Value:	1Bh
		Format:	Opcode
	Untyped Atomic Float Operation message		
13	Return Data Control		
		Default Value:	1h
		Format:	Opcode
	Specifies that return data is sent back to the thread.		
12	SIMD Mode		
	Format:	MDC_SM2R	
	Specifies the SIMD mode of the message (number of slots processed)		
11	Data Width		
		Default Value:	0h
		Format:	Opcode
	Operations are on 32-bit floats.		
10:8	Atomic Float Operation		
	Format:	MDC_FOP	
	Specifies the atomic float operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS_SLM_A32	
	Specifies the Binding Table Index for the message		

Dword Untyped Atomic Float Write Only Operation MSD

MSD1W_DWAF - Dword Untyped Atomic Float Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type

MSD1W_DWAF - Dword Untyped Atomic Float Write Only Operation MSD

		Default Value:	1Bh
		Format:	Opcode
	Untyped Atomic Float Operation message		
13	Return Data Control		
		Default Value:	0h
		Format:	Opcode
	Specifies that no return data is sent back to the thread.		
12	SIMD Mode		
	Format:	MDC_SM2R	
	Specifies the SIMD mode of the message (number of slots processed)		
11	Data Width		
		Default Value:	0h
		Format:	Opcode
	Operations are on 32-bit floats.		
10:8	Atomic Float Operation		
	Format:	MDC_FOP	
	Specifies the atomic float operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS_SLM_A32	
	Specifies the Binding Table Index for the message		

Dword Untyped Atomic Integer with Return Data Operation MSD

MSD1R_DWAI - Dword Untyped Atomic Integer with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type

MSD1R_DWAI - Dword Untyped Atomic Integer with Return Data Operation MSD

		Default Value:	02h
		Format:	Opcode
	Untyped Atomic Integer Operation message		
13	Return Data Control		
		Default Value:	1h
		Format:	Opcode
	Specifies that return data is sent back to the thread.		
12	SIMD Mode		
	Format:	MDC_SM2R	
	Specifies the SIMD mode of the message (number of slots processed)		
11:8	Atomic Integer Operation		
	Format:	MDC_AOP	
	Specifies the atomic integer operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS_SLM_A32	
	Specifies the Binding Table Index for the message		

Dword Untyped Atomic Integer Write Only Operation MSD

MSD1W_DWAI - Dword Untyped Atomic Integer Write Only Operation MSD								
Source:		EuSubFunctionDataPort1						
Length Bias:		1						
DWord	Bit	Description						
0	31	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	30	Packed Data Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td style="width: 30%;">0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF).</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Restriction</th> </tr> <tr> <td style="text-align: center;">Only 32-bit data packing is supported at this time.</td> </tr> </table>	Default Value:	0 32 bit	Format:	Enable	Restriction	Only 32-bit data packing is supported at this time.
	Default Value:	0 32 bit						
	Format:	Enable						
	Restriction							
	Only 32-bit data packing is supported at this time.							
	29	Packed Address Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td style="width: 30%;">0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.</p>	Default Value:	0 32 bit	Format:	Enable		
Default Value:	0 32 bit							
Format:	Enable							
28:25	Message Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4					
Format:	U4							
24:20	Response Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5					
Format:	U5							
19	Header Present <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MDC_MHP</td> </tr> </table> <p>If set, indicates that the message includes the header.</p>	Format:	MDC_MHP					
Format:	MDC_MHP							
18:14	Message Type							

MSD1W_DWAI - Dword Untyped Atomic Integer Write Only Operation MSD

		Default Value:	02h
		Format:	Opcode
	Untyped Atomic Integer Operation message		
13	Return Data Control		
		Default Value:	0h
		Format:	Opcode
	Specifies that no return data is sent back to the thread.		
12	SIMD Mode		
	Format:	MDC_SM2R	
	Specifies the SIMD mode of the message (number of slots processed)		
11:8	Atomic Integer Operation		
	Format:	MDC_AOP	
	Specifies the atomic integer operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS_SLM_A32	
	Specifies the Binding Table Index for the message		

Else

else - Else				
Source:	Eulsa			
Length Bias:	4			
<p>The else instruction is an optional statement within an if/else/endif block of code. It restricts execution within the else/endif portion to the opposite set of channels enabled under the if/else portion. Channels which were inactive prior to entering the if/endif block remain inactive throughout the entire block. All enabled channels upon arriving the else instruction will be redirected to the matching endif. If all channels are redirected (by else or before else), a relative jump is performed to the location specified by <JIP>. The jump target should be the matching endif instruction for that conditional block. The following table describes the 32-bit <JIP>. In instruction binary, <JIP> is at location <src1> and must be of type D (signed dword integer). <JIP> must be an immediate operand, it is a signed 32-bit number and is intended to be forward referencing. This value is added to IP pre-increment. If the <branch_ctrl> bit is set, then the <JIP> points to the first join instruction within the else block and <UIP> points to the endif instruction. If the <branch_ctrl> bit is not set, <JIP> and <UIP>, both point to endif.</p>				
Format:	<pre>else (exec_size) JIP UIP branch_ctrl</pre>			
Restriction				
Predication is not allowed.				
The execution size must be the same for the if, else, and endif instructions of the same code block.				
The branch_ctrl must be set equal to (JIP != UIP).				
Syntax				
else (exec_size) imm32 imm32 branch_ctrl				
Pseudocode				
<pre>Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn.channel[n] == 1 branch_ctrl) { PcIP[n] = IP + JIP; } else { PcIP[n] = IP + UIP; } } if (PcIP != (IP+1)) { // for all channels Jump(IP + JIP); }</pre>				
Predication	Conditional Modifier	Saturation	Source Modifier	Syntax
N	N	N	N	Unspecified
DWord	Bit	Description		
0..3	127:96	Reserved		
		Exists If:	([Src0.IsImm]==false)	
		Format:	MBZ	
	127:96	JIP		
Exists If:		([Src0.IsImm]==true)		
		Format:	S31	

else - Else							
	The byte-aligned jump distance if a jump is taken for the channel.						
95:80	Reserved <table border="1"> <tr> <td>Exists If:</td> <td>$([Src0.IsImm] == false)$</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	$([Src0.IsImm] == false)$	Format:	MBZ		
Exists If:	$([Src0.IsImm] == false)$						
Format:	MBZ						
95:64	Reserved <table border="1"> <tr> <td>Exists If:</td> <td>$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == false)$</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == false)$	Format:	MBZ		
Exists If:	$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == false)$						
Format:	MBZ						
95:64	UIP <table border="1"> <tr> <td>Exists If:</td> <td>$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == true)$</td> </tr> <tr> <td>Format:</td> <td>S31</td> </tr> </table> <p>The byte aligned jump distance if a jump is taken for the instruction.</p>	Exists If:	$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == true)$	Format:	S31		
Exists If:	$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == true)$						
Format:	S31						
79:66	Src0.Operand <table border="1"> <tr> <td>Exists If:</td> <td>$([Src0.IsImm] == false)$</td> </tr> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Exists If:	$([Src0.IsImm] == false)$	Format:	DirectOperand		
Exists If:	$([Src0.IsImm] == false)$						
Format:	DirectOperand						
65:64	Reserved <table border="1"> <tr> <td>Exists If:</td> <td>$([Src0.IsImm] == false)$</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	$([Src0.IsImm] == false)$	Format:	MBZ		
Exists If:	$([Src0.IsImm] == false)$						
Format:	MBZ						
63:50	Dst.Operand <table border="1"> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Format:	DirectOperand				
Format:	DirectOperand						
49:48	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
47	Src1.IsImm This field indicate that Source 1 operand is carrying an immediate value <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>false</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </tbody> </table>	Value	Name	0	false	1	true
Value	Name						
0	false						
1	true						
46	Src0.IsImm This field indicate that Source 0 operand is carrying an immediate value <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>false</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </tbody> </table>	Value	Name	0	false	1	true
Value	Name						
0	false						
1	true						
45:34	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
33	BranchCtrl This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.						

else - Else

32	AtomicCtrl Format: AtomicCtrl										
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".										
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.									
30	Reserved										
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.										
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.	
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields										
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.	
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.										

else - Else

23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

End If

endif - End If				
Source:	Eulsa			
Length Bias:	4			
<p>The endif instruction terminates an if/else/endif block of code. It restores execution to the channels that were active prior to the if/else/endif block. The endif instruction is also used to hop out of nested conditionals by jumping to the end of the next outer conditional block when all channels are disabled.</p> <p>The following table describes the 32-bit JIP. In instruction binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand, it is a signed 32-bit number. This value is added to IP pre-increment.</p>				
Format:	endif JIP			
Restriction				
Predication is not allowed.				
The execution size must be the same for the if, else, and endif instructions of the same code block.				
Syntax				
endif (exec_size) imm32				
Pseudocode				
<pre>Evaluate (WrEn); if (WrEn == 0) { // all channels false Jump (IP + JIP); }</pre>				
Predication	Conditional Modifier	Saturation	Source Modifier	
N	N	N	N	
DWord	Bit	Description		
0..3	127:96	Reserved		
		Exists If:	([Src0.IsImm]==false)	
		Format:	MBZ	
	127:96	JIP		
		Exists If:	([Src0.IsImm]==true)	
		Format:	S31	
			The byte-aligned jump distance if a jump is taken for the channel	
	95:80	Reserved		
		Exists If:	([Src0.IsImm]==false)	
Format:		MBZ		
95:64	Reserved			

endif - End If

		Exists If:	([Src0.IsImm]==true)
		Format:	MBZ
79:66	Src0.Operand		
		Exists If:	([Src0.IsImm]==false)
		Format:	DirectOperand
65:64	Reserved		
		Exists If:	([Src0.IsImm]==false)
		Format:	MBZ
63:50	Dst.Operand		
		Format:	DirectOperand
49:47	Reserved		
		Format:	MBZ
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
	1	true	
45:34	Reserved		
		Format:	MBZ
33	BranchCtrl		
	This field is used by <i>goto</i> , <i>if</i> , and <i>elsei</i> nstructions to control branching. See the goto instruction description for more information about BranchCtrl.		
32	AtomicCtrl		
		Format:	AtomicCtrl
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
		Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for		

endif - End If

		use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv	<p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	PredCtrl	<table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl										
23	FlagRegNum[0]	This field specifies bit[0] of the register number for a flag register operand.									
22	FlagSubRegNum	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.									
21:19	ChanOff	<table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff										
18:16	ExecSize										

endif - End If						
		<table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> <tr> <td colspan="2">This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</td> </tr> </table>	Format:	ExecSize	This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.	
Format:	ExecSize					
This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.						
	15:0	<table border="1"> <tr> <td colspan="2">Header</td> </tr> <tr> <td>Format:</td> <td>Header</td> </tr> </table>	Header		Format:	Header
Header						
Format:	Header					

Extended Math Function

math - Extended Math Function	
Source:	Eulsa
Length Bias:	4
<p>The math instruction performs extended math function on the components in src0, or src0 and src1, and write the output to the channels of dst. The type of extended math function are based on the FC[3:0] encoding in the table below.</p>	
Format:	<code>[(pred)] math.<FC> (exec_size) dst src0 src1</code>
Restriction	
Accumulator access is allowed only for IEEE macro functions (invn and rsqtm).	
The math instruction does not support indirect addressing modes.	
The only supported rounding mode for math instruction is Round to Nearest Even.	
INT DIV function does not support SIMD16. INT DIV function does not support simultaneous write to two registers. INT DIV function does not support source modifiers.	
The FDIV function is not supported in ALT_MODE.	
The math instruction can use GRF or immediates as source. The source formats for immediates must be one of the source formats supported by math operation.	
DepCtrl must not be used.	
The math instruction must use GRF as destination.	
The supported regioning mode for math instruction is align1 and align16. The following restrictions apply for align1 mode: Scalar source is supported. Source and destination horizontal stride must be the same. Regioning must ensure $\text{Src.Vstride} = \text{Src.Width} * \text{Src.Hstride}$. Source and destination offset must be the same, except the case of scalar source.	
Half-float denorms are always retained.	
Math Operation rules when float and half-floats are mixed between source or between source and destination operands. The half-float operand must be interleaved in the register for align1 and the source and destination register offset must be the same to DW granularity. For align16, the half-float operand is allowed to be packed.	
The execution size must be no more than 8 when half-floats are used in source or destination operand.	
<p>The source operand must not span two registers if the destination operand spans just one register Example:</p> <p>Case (a) // Allowed. The source must be strided by 2. the offset is allowed to select between lower/upper 16b <code>math.invm (8) r10:f r11.0<16;8,2>:hf math.invm (8) r10:f r11.1<16;8,2>:hf math.invm (8) r10:f r11.0<16;8,2>:hf r12.1<16;8,2>:hf</code> Case (b) // Allowed. The destination must be strided by 2. The offset is allowed to select between lower/upper 16b <code>math.invm (8) r10.0<2>:hf r11.0<8;8,1>:f math.invm (8) r10.1<2>:hf r11.0<8;8,1>:f math.invm (8) r10.0<2>:hf r11.0<16;8,2>:hf r12.0<16;8,2>:hf</code> Case (c) // Allowed. Destination has stride of 2. The offset is allowed to select between upper/lower 16b <code>math.invm (8) r10.0<2>:hf r11.0<8;8,1>:f r12.1<16;8,2>:hf math.invm (8) r10.1<2>:hf r11.1<16;8,2>:hf r12.0<8;8,1>:f</code> Case (d) // Not Allowed. Destination is half-float but is not interleaved. <code>math.invm (8) r10.0<1>:hf r11.0<8;8,1>:f</code> Case (e) // Not Allowed. Source is</p>	

math - Extended Math Function

half-float but not interleaved math.invm (8) r10.0<2>:hf r11.0<8;8,1>:f r12.0<8;8,1>:hf Case (f) // Not Allowed. Source operand spans 2 registers while destination spans one register. math.sin (8) r83.8<1>:hf r12.4<4;4,1>:f

Math Operation rules when half-floats are used on both source and destination operands. The execution size must be 8. The half-float source must be packed or interleaved. When interleaving, both source and destination must be interleaved. Example: Case (a) // Allowed. The source and destination are packed or interleaved
 math.invm (8) r10.0:hf r11.0<8;8,1>:hf math.invm (8) r10.0<2>:hf r11.0<16;8,2>:hf math.invm (8) r10.8:hf r11.0<8;8,1>:hf math.invm (8) r10.8<2>:hf r11.0<16;8,2>:hf

For one source math operations src1 must encode the null register.

Syntax

[(pred)] math.<FC> (exec_size) reg reg reg

Pseudocode

```
Evaluate (WrEn); for (n = 0; n < exec_size; n++) {
  if (WrEn.channel[n] == 1) {
    switch FC[3:0] {
      case 1h: // math.invm          dst.channel[n] =
rcp(src0.channel[n]);              case 2h: // math.log          dst.channel[n] =
log(src0.channel[n]);              case 3h: // math.exp          dst.channel[n] =
exp(src0.channel[n]);              case 4h: // math.sqrt         dst.channel[n] =
sqrt(src0.channel[n]);            case 5h: // math.rsqrt        dst.channel[n] =
rsqrt(src0.channel[n]);           case 6h: // math.sin         dst.channel[n] =
sin(src0.channel[n]);             case 7h: // math.cos         dst.channel[n] =
cos(src0.channel[n]);             case 9h: // math.fdiv (src0 / src1)
dst.channel[n] = fdiv(src0.channel[n], src1.channel[n]);
dst.channel[n] = pow(src0.channel[n], src1/channel[n]);
(src0 / src1)                       case Ah: // math.pow
idiv(src0.channel[n], src1.channel[n]);
dst.channel[n] = quotient;          dst+1.channel[n] = remainder;
Ch: // math.iqot                    idiv(src0.channel[n], src1.channel[n]);
dst.channel[n] = quotient;          case Dh: // math.irem
idiv(src0.channel[n], src1.channel[n]);
dst.channel[n] = remainder;
case Eh: // math.invm              dst.channel[n] = invm(src0.channel[n],
src1.channel[n]);                  case Fh: // math.rsqtm
rsqtm(src0.channel[n]);            } } } }
```

Predication	Conditional Modifier	Saturation	Source Modifier	Syntax	Subfunctions []
Y	N	Y	Y	Unspecified	Unspecified

Src Types	Dst Types
D	D
UD	UD
F, HF	F, HF

DWord	Bit	Description
0..3	127:126	Reserved
		Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false)
		Format: MBZ
	127:96	Src0.ImmValue[31:0]
		Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false)

math - Extended Math Function

127:96	Src1.ImmValue[31:0]		
	Exists If:	(([Src0.IsImm]==false) AND ([Src1.IsImm]==true))	
	Reserved		
	Exists If:	(([Src0.IsImm]==false) AND ([Src1.IsImm]==false))	
	Format:	MBZ	
	121:120	Src1.Mod	
		Format:	SrcMod
	119:116	Src1.VertStride	
		Format:	VertStride
	115:113	Src1.Width	
		Format:	Width
	112	Reserved	
		Format:	MBZ
	111:98	Src1.Operand	
Format:		DirectOperand	
97:96	Src1.HorzStride		
	Format:	HorzStride	
95:92	FuncCtrl		
	Format:	MathFC	
91:88	Src1.DataType		
	Format:	RegDataType	
91:88	Src1.DataType		
	Format:	ImmDataType	
91:64	Reserved		
	Format:	MBZ	
87:84	Src0.VertStride		
	Exists If:	(([Src0.IsImm]==false))	

math - Extended Math Function		
	Format:	VertStride
83:81	Src0.Width	
	Exists If:	(([Src0.IsImm]==false)
	Format:	Width
80	Reserved	
	Exists If:	(([Src0.IsImm]==false)
	Format:	MBZ
79:66	Src0.Operand	
	Exists If:	(([Src0.IsImm]==false) AND (([FuncCtrl]!=INVM) AND ([FuncCtrl]!=RSQTM))
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	(([Src0.IsImm]==false) AND (([FuncCtrl]==INVM) OR ([FuncCtrl]==RSQTM))
	Format:	MacroOperand
65:64	Src0.HorzStride	
	Exists If:	(([Src0.IsImm]==false)
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	(([FuncCtrl]==INVM) OR ([FuncCtrl]==RSQTM)
	Format:	MacroOperand
63:50	Dst.Operand	
	Exists If:	(([FuncCtrl]!=INVM) AND ([FuncCtrl]!=RSQTM)
	Format:	DirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value	
	Value	Name
	0	false
1	true	
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false
1	true	
45:44	Reserved	
	Exists If:	(([Src0.IsImm]==true)

math - Extended Math Function		
	Format:	MBZ
45:44	Src0.Mod	
	Exists If:	((Src0.IsImm) == false)
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Reserved	
	Format:	MBZ
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name Description
	0	Normal [Default] Normal. Per channel write enable used for final write enable generation.
1	NoMask NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name Description

math - Extended Math Function					
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.		
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.		
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields				
	Value	Name	Description		
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.		
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.		
27:24	PredCtrl Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="text-align: center;">PredCtrl</td></tr></table> This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.				PredCtrl
	PredCtrl				
23	FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.				
22	FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.				
21:19	ChanOff Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="text-align: center;">ChanOff</td></tr></table> This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.				ChanOff
	ChanOff				
18:16	ExecSize Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="text-align: center;">ExecSize</td></tr></table> This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.				ExecSize
	ExecSize				
15:0	Header Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="text-align: center;">Header</td></tr></table>				Header
	Header				

Find First Bit from LSB Side

fbl - Find First Bit from LSB Side			
Source:	Eulsa		
Length Bias:	4		
The fbl instruction counts component-wise the number of LSB 0 bits before the first 1 bit in src0, storing that number in dst.			
Format:	<code>[(pred)] fbl (exec_size) dst src0</code>		
Programming Notes			
If src0 contains no 1 bits, store 0xFFFFFFFF in dst.			
Restriction			
No accumulator access, implicit or explicit.			
Syntax			
<code>[(pred)] fbl (exec_size) reg reg [(pred)] fbl (exec_size) reg imm32</code>			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD cnt = 0; UD udScalar = src0.chan[n]; while ((udScalar & 1) == 0 && cnt != 32) { cnt ++; udScalar = udScalar >> 1; } if (src0.chan[n] == 0x00000000) { dst.chan[n] = 0xFFFFFFFF; } else { dst.chan[n] = cnt; } } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
Src Types	Dst Types		
UD	UD		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0] Exists If: <code>((Src0.IsImm)==true)</code>	
	95:92	CondCtrl Exists If: <code>((Src0.IsImm)==false) OR (((Src0.DataType)!=:q) AND ((Src0.DataType)!=:uq) AND ((Src0.DataType)!=:df))</code> Format: FlagModifier	
	95:64	Src0.ImmValue[63:32] Exists If: <code>((Src0.IsImm)==true) AND (((Src0.DataType)==:q) OR ((Src0.DataType)==:uq) OR ((Src0.DataType)==:df))</code>	

fbl - Find First Bit from LSB Side

87:84	Src0.VertStride		
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))	
	Format:	VertStride	
	83:81	Src0.Width	
		Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
		Format:	Width
	80	Src0.AddrMode	
		Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
		Format:	AddrMode
	79:66	Src0.Operand	
		Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)
		Format:	DirectOperand
79:66	Src0.Operand		
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	

fbl - Find First Bit from LSB Side

	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name Description
	0	Normal [Default] Normal. Per channel write enable used for final write enable generation.
	1	NoMask NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name Description

fbf - Find First Bit from LSB Side												
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	<p>PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>			Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>			Format:	PredCtrl							
Format:	PredCtrl											
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>											
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>											
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>			Format:	ChanOff							
Format:	ChanOff											
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>			Format:	ExecSize							
Format:	ExecSize											
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>			Format:	Header							
Format:	Header											

Find First Bit from MSB Side

fbh - Find First Bit from MSB Side			
Source:	Eulsa		
Length Bias:	4		
<p>If src0 is unsigned, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and positive, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and negative, the fbh instruction counts component-wise the leading ones from src0 and stores the resulting counts in dst.</p>			
Format:	[(pred)] fbh (exec_size) dst src0		
Programming Notes			
If src0 is zero, store 0xFFFFFFFF in dst.			
If src0 is signed and is -1 (0xFFFFFFFF), store 0xFFFFFFFF in dst.			
Restriction			
No accumulator access, implicit or explicit.			
Syntax			
[(pred)] fbh (exec_size) reg reg [(pred)] fbh (exec_size) reg imm32			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD cnt = 0; if (src0 is unsigned) { UD udScalar = src0.chan[n]; while ((udScalar & (1 << 31)) == 0 && cnt != 32) { cnt ++; udScalar = udScalar << 1; } if (src0.chan[n] == 0x00000000) { dst.chan[n] = 0xFFFFFFFF; } } else { dst.chan[n] = cnt; } } else { // src0 is signed. D dScalar = src0.chan[n]; bit cval = dScalar[31]; while ((dScalar & (1 << 31)) == cval && cnt != 32) { cnt ++; dScalar = dScalar << 1; } if ((src0.chan[n] == 0xFFFFFFFF) (src0.chan[n] == 0x00000000)) { dst.chan[n] = 0xFFFFFFFF; } } else { dst.chan[n] = cnt; } } } } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
Src Types	Dst Types		
*D	UD		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0]	
		Exists If:	[(Src0.IsImm) == true]

fbh - Find First Bit from MSB Side

95:92	CondCtrl				
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))			
	Format:	FlagModifier			
	95:64	Src0.ImmValue[63:32]			
		Exists If:	(([Src0.IsImm]==true) AND ((([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df)))		
		87:84	Src0.VertStride		
			Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))	
			Format:	VertStride	
			83:81	Src0.Width	
				Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
Format:				Width	
80				Src0.AddrMode	
				Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
	Format:			AddrMode	
	79:66			Src0.Operand	
		Exists If:		((([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct))	
		Format:		DirectOperand	
		79:66		Src0.Operand	
			Exists If:	((([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect))	
			Format:	IndirectOperand	
			65:64	Src0.HorzStride	
Exists If:				(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))	
Format:				HorzStride	
63:50				Dst.Operand	
	Exists If:			(([Dst.AddrMode]==Indirect)	
	Format:			IndirectOperand	
	63:50			Dst.Operand	
		Exists If:		(([Dst.AddrMode]==Direct)	
		Format:		DirectOperand	
		49:48		Dst.HorzStride	

fbh - Find First Bit from MSB Side

	Format:	HorzStride
47	Reserved	
	Format:	MBZ
46	Src0.IsImm This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name Description
	0	Normal [Default] Normal. Per channel write enable used for final write enable generation.
	1	NoMask NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ

fbh - Find First Bit from MSB Side

	<p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.								
28	<p>PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									

fbh - Find First Bit from MSB Side				
	18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
	Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">Header</td> </tr> </table>	Format:	Header	
Format:	Header			

Fraction

frc - Fraction			
Source:	Eulsa		
Length Bias:	4		
<p>The frc instruction computes, component-wise, the truncate-to-minus-infinity fractional values of src0 and stores the results in dst. The results, in the range of [0.0, 1.0], are the fractional portion of the source data. The result is in the range [0.0, 1.0] irrespective of the rounding mode. Floating-point fraction computation follows the rules in the following tables, based on the current floating-point mode.</p>			
Format:	[(pred)] frc[.cmod] (exec_size) dst src0		
Syntax			
[(pred)] frc[.cmod] (exec_size) reg reg [(pred)] frc[.cmod] (exec_size) reg imm32			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] - floor(src0.chan[n]); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y
Src Types	Dst Types		
F	F		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0]	
		Exists If:	((Src0.IsImm)==true)
	95:92	CondCtrl	
		Exists If:	((Src0.IsImm)==false) OR (((Src0.DataType)!=:q) AND ((Src0.DataType)!=:uq) AND ((Src0.DataType)!=:df))
		Format:	FlagModifier
	95:64	Src0.ImmValue[63:32]	
		Exists If:	((Src0.IsImm)==true) AND (((Src0.DataType)==:q) OR ((Src0.DataType)==:uq) OR ((Src0.DataType)==:df))
	87:84	Src0.VertStride	
		Exists If:	((Src0.IsImm)==false) OR (((Src0.DataType)!=:q) AND ((Src0.DataType)!=:uq) AND ((Src0.DataType)!=:df))
		Format:	VertStride
	83:81	Src0.Width	

frc - Fraction

	Exists If:	([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))	
	Format:	Width	
80	Src0.AddrMode		
	Exists If:	([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))	
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))) AND ([Src0.AddrMode] == Direct))	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))) AND ([Src0.AddrMode] == Indirect))	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df))	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode] == Indirect)	
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode] == Direct)	
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		

frc - Fraction

		Exists If:	([Src0.IsImm]==false)
		Format:	RegDataType
43:40	Src0.DataType		
		Exists If:	([Src0.IsImm]==true)
		Format:	ImmDataType
39:36	Dst.DataType		
		Format:	RegDataType
35	Dst.AddrMode		
		Format:	AddrMode
34	Saturate		
		Format:	Saturate
33	AccWrCtrl		
		Format:	AccWrCtrl
32	AtomicCtrl		
		Format:	AtomicCtrl
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
		Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask		

frc - Fraction

	<p>for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize							
Format:	ExecSize									
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>	Format:	Header							
Format:	Header									

Goto

goto - Goto			
Source:	Eulsa		
Length Bias:	4		
<p>The goto instruction directs the instruction pointer to the offset specified by the UIP offset or to the next IP based on the BranchCtrl bit in the instruction. The active channels that are predicated on this instruction will take the IP + UIP path when BranchCtrl is set else the channels take IP + 1. The active channels that are not predicated on this instruction will be made inactive and waiting to be joined at the join IP. The join IP is IP + UIP when BranchCtrl is clear else it is the next IP.</p> <p>When there are no active channels the instruction pointer will move to IP + JIP.</p> <p>The goto instruction is used in conjunction with a join instruction. A goto deactivates some channels that are reactivated at some program-specified join instruction. See the join instruction for the activation rules.</p> <p>The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence.</p> <p>The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer).</p> <p>If SPF is ON, none of the PcIP are updated.</p>			
Format:	<pre>[(pred)] goto (exec_size) JIP UIP branch_ctrl</pre>		
Restriction			
Cannot have a goto in the body and the corresponding join in the function or the subroutine. Similarly the brd and brc.			
Syntax			
<pre>[(pred)] goto (exec_size) imm32 imm32 branch_ctrl</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { // for the predicated active channels PcIP[n] = IP + UIP; } else { // join IP, for the active non predicated channels PcIP[n] = IP + 1; } } if (BranchCtrl) { if (PcIP != (IP + UIP)) { // for all channels if (PcIP != (IP + 1)) { // for all channels Jump(IP + JIP); } else { Jump(IP + 1); } } else { Jump(IP + UIP); } } else { if (PcIP != (IP + 1)) { // for all channels Jump(IP + JIP); } else { Jump(IP + 1); } } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	

goto - Goto							
	<table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	MBZ		
Exists If:	([Src0.IsImm]==false)						
Format:	MBZ						
127:96	<p>JIP</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==true)</td> </tr> <tr> <td>Format:</td> <td>S31</td> </tr> </table> <p>The byte-aligned jump distance if a jump is taken for the channel.</p>	Exists If:	([Src0.IsImm]==true)	Format:	S31		
Exists If:	([Src0.IsImm]==true)						
Format:	S31						
95:80	<p>Reserved</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	MBZ		
Exists If:	([Src0.IsImm]==false)						
Format:	MBZ						
95:64	<p>Reserved</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==true) AND ([Src1.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)	Format:	MBZ		
Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)						
Format:	MBZ						
95:64	<p>UIP</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==true) AND ([Src1.IsImm]==true)</td> </tr> <tr> <td>Format:</td> <td>S31</td> </tr> </table> <p>The byte aligned jump distance if a jump is taken for the instruction.</p>	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)	Format:	S31		
Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)						
Format:	S31						
79:66	<p>Src0.Operand</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	DirectOperand		
Exists If:	([Src0.IsImm]==false)						
Format:	DirectOperand						
65:64	<p>Reserved</p> <table border="1"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	MBZ		
Exists If:	([Src0.IsImm]==false)						
Format:	MBZ						
63:50	<p>Dst.Operand</p> <table border="1"> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Format:	DirectOperand				
Format:	DirectOperand						
49:48	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
47	<p>Src1.IsImm</p> <p>This field indicate that Source 1 operand is carrying an immediate value</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>false</td> </tr> <tr> <td>1</td> <td>true</td> </tr> </tbody> </table>	Value	Name	0	false	1	true
Value	Name						
0	false						
1	true						
46	<p>Src0.IsImm</p> <p>This field indicate that Source 0 operand is carrying an immediate value</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>false</td> </tr> </tbody> </table>	Value	Name	0	false		
Value	Name						
0	false						

goto - Goto

	1	true
45:34	Reserved	
	Format:	MBZ
33	BranchCtrl This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name Description
	0	Normal [Default] Normal. Per channel write enable used for final write enable generation.
	1	NoMask NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name Description
	0	NoCompaction [Default] No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name Description
	0	Positive [Default] Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

goto - Goto

27:24	<p>PredCtrl</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl
Format:	PredCtrl		
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

GPGPU_CSR_BASE_ADDRESS

GPGPU_CSR_BASE_ADDRESS			
Source:	BSpec		
Length Bias:	2		
The GPGPU_CSR_BASE_ADDRESS command sets the base pointers for EU and L3 to Context Save and Restore EU State and SLM for GPGPU mid-thread preemption.			
Programming Notes			
Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance. State and instruction caches are flushed on completion of the flush.			
SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set prior to programming GPGPU_CSR_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	Opcode
	28:27	Command SubType	
		Default Value:	0h GFXPIPE_COMMON
		Format:	Opcode
	26:24	3D Command Opcode	
Default Value:		1h GFXPIPE_NONPIPELINED	
Format:		Opcode	
23:16	3D Command Sub Opcode		
	Default Value:	04h GPGPU_CSR_BASE_ADDRESS	
	Format:	Opcode	
15:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Format:	=n	
	Value	Name	Description
	1h	Length_1 [Default]	Excludes DWord(0,1)
1..2	63:12	GPGPU CSR Base Address	
		Format:	GraphicsAddress[63:12] Specifies the 256K-byte aligned base address for GPGPU context GraphicsAddress [63:48] are

GPGPU_CSR_BASE_ADDRESS	
	ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
11:0	Reserved
	Format: MBZ

GPGPU_WALKER

GPGPU_WALKER		
Source:	RenderCS	
Length Bias:	2	
Programming Notes		
<p>If the threads spawned by this command are required to observe memory writes performed by threads spawned from a previous command, software must precede this command with a command that performs a memory flush (e.g., MI_FLUSH).</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE Format: OpCode
	28:27	Pipeline
		Default Value: 2h Media Format: OpCode
	26:24	Media Command Opcode
		Default Value: 1h GPGPU_WALKER Format: OpCode
	23:16	SubOpcode
		Default Value: 05h GPGPU_WALKER SubOp Format: OpCode
	15:11	Reserved
		Format: MBZ
10	Indirect Parameter Enable	
	Format: Enable	
<p>If set, the values in DW 7, 10, 12 are ignored and replaced by the current values of the corresponding GPGPU_xxx MMIO registers:</p> <ul style="list-style-type: none"> • GPGPU_DISPATCHDIMX (instead of DW7) • GPGPU_DISPATCHDIMY (instead of DW10) • GPGPU_DISPATCHDIMZ (instead of DW12) 		
9	Reserved	
	Format: MBZ	
8	Predicate Enable	
	Format: Enable	
<p>If set, this command is executed (or not) depending on the current value of the MI Predicate</p>		

GPGPU_WALKER										
		internal state bit. This command is ignored only if PredicateEnable is set and the Predicate state bit is 0.								
	7:0	<p>DWord Length</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 40%;">Description</th> </tr> </thead> <tbody> <tr> <td>0Dh</td> <td>DWORD_COUNT_n [Default]</td> <td>Excludes DWord (0,1)</td> </tr> </tbody> </table>	Format:	=n	Value	Name	Description	0Dh	DWORD_COUNT_n [Default]	Excludes DWord (0,1)
Format:	=n									
Value	Name	Description								
0Dh	DWORD_COUNT_n [Default]	Excludes DWord (0,1)								
1	31:8	Reserved								
	7:6	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
5:0	<p>Interface Descriptor Offset</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.</p>	Format:	U6							
Format:	U6									
2	31:17	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
16:0	<p>Indirect Data Length</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U17</td> </tr> </table> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. If Indirect Data is enabled, it is assumed that CURBE is not being used. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned.</p>	Format:	U17							
Format:	U17									
3	31:6	<p>Indirect Data Start Address</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>IndirectObjectOffset[31:6]</td> </tr> </table> <p>This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the Indirect Object Base Address. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation. It is the 64-byte aligned address of the indirect data.</p> <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0x0,0x3FFFFFF]</td> <td></td> <td>Range 0 .. 512 MB. (Bits 31:29 MBZ)</td> </tr> </tbody> </table>	Format:	IndirectObjectOffset[31:6]	Value	Name	Description	[0x0,0x3FFFFFF]		Range 0 .. 512 MB. (Bits 31:29 MBZ)
		Format:	IndirectObjectOffset[31:6]							
	Value	Name	Description							
[0x0,0x3FFFFFF]		Range 0 .. 512 MB. (Bits 31:29 MBZ)								
5:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
4	31:30	<p>SIMD Size</p> <p>This field determines the size of the payload and the number of bits of the execution mask that are expected. The kernel pointed to by the interface descriptor should match the SIMD declared here.</p> <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Value	Name	Description					
Value	Name	Description								

GPGPU_WALKER		
	0	SIMD8 8 LSBs of the execution mask are used
	1	SIMD16 16 LSBs used in execution mask
	2	SIMD32 32 bits of execution mask used
29:22	Reserved	
	Format:	MBZ
21:16	Thread Depth Counter Maximum The maximum value of the thread depth counter. Since the counter starts at 0, the depth is this value + 1. (Thread_Depth_Max+1)*(Thread_Height_Max+1)*(Thread_Width_Max+1) must equal Number of Threads in GPGPU Thread Group in the Interface Descriptor.	
15:14	Reserved	
	Format:	MBZ
13:8	Thread Height Counter Maximum	
	Format:	U6-1
	The maximum value of the thread height counter. The height is this value + 1.	
7:6	Reserved	
	Format:	MBZ
5:0	Thread Width Counter Maximum	
	Format:	U6-1
	The maximum value of the thread width counter. The width is this value + 1.	
5	31:0	Thread Group ID Starting X This is the initial value of the X component of the thread group. When X reaches the maximum value it rolls around to this value.
6	31:0	Reserved
	Format:	MBZ
7	31:0	Thread Group ID X Dimension The X dimension of the thread group (maximum X is dimension -1)
8	31:0	Thread Group ID Starting Y This is the initial value of the Y component of the thread group. When Y reaches the maximum value it rolls around to this value.
9	31:0	Reserved
	Format:	MBZ
10	31:0	Thread Group ID Y Dimension The Y dimension of the thread group (maximum Y is dimension -1)
11	31:0	Thread Group ID Starting/Resume Z This is the initial value of the Z component of the thread group.
12	31:0	Thread Group ID Z Dimension

GPGPU_WALKER		
		The Z dimension of the thread group (maximum Z is dimension -1)
13	31:0	Right Execution Mask The execution mask used for threads on the right (maximum X) of the thread group.
14	31:0	Bottom Execution Mask The execution mask used for threads on the bottom (maximum Y) of the thread group.

Half Precision HI8DS Render Target Write MSD

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 1h Format: Opcode Half precision data message
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. Programming Notes This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	Message Type

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

		Default Value:	0Ch
		Format:	Opcode
	Render Target Write message		
13	Per-Sample PS Enable		
		Format:	Enable
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.		
	When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.		
	When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).		
	When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.		
12	Last Render Target Select		
		Format:	Enable
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select		
		Format:	MDC_RT_SGS
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype		
		Default Value:	3h
		Format:	Opcode
	SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].		
7:0	Binding Table Index		
		Format:	MDC_BTS
	Specifies the Binding Table Index for the message		



Half Precision LO8DS Render Target Write MSD

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 1h Format: Opcode Half precision data message
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. Programming Notes This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	Message Type

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

		Default Value:	0Ch
		Format:	Opcode
	Render Target Write message		
13	Per-Sample PS Enable		
	Format:	Enable	
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.		
	When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_CO as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.		
	When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).		
	When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.		
12	Last Render Target Select		
	Format:	Enable	
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select		
	Format:	MDC_RT_SGS	
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype		
	Default Value:	2h	
	Format:	Opcode	
	SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].		

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

	7:0	Binding Table Index		
		<table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%; text-align: center;">MDC_BT</td> </tr> </table>	Format:	MDC_BT
Format:	MDC_BT			
		Specifies the Binding Table Index for the message		

Half Precision REP16 Render Target Write MSD

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 1h Format: Opcode Half precision data message
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. Programming Notes This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	Message Type

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

		Default Value:	0Ch
		Format:	Opcode
	Render Target Write message		
13	Per-Sample PS Enable		
	Format:	Enable	
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.		
	When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.		
	When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).		
	When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.		
12	Last Render Target Select		
	Format:	Enable	
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select		
	Format:	MDC_RT_SGS	
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype		
	Default Value:	1h	
	Format:	Opcode	
	SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is		

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

		selected, slots [31:16] are referenced instead of [15:0].
	7:0	Binding Table Index Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision SIMD8 Render Target Write MSD

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved
		Format: MBZ
	30	Message Precision Subtype
		Default Value: 1h
		Format: Opcode
	Half precision data message	
	29	Reserved
		Format: MBZ
	28:25	Message Length
		Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
24:20	Response Length	
	Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.	
19	Header Present	
	Format: MDC_MHP If set, indicates that the message includes the 2-register header.	
18	Per-Coarse Pixel PS outputs enable	
	Format: Enable	
	This bit indicates the render target write is a coarse pixel write.	
Programming Notes		
This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.		
17:14	Message Type	

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

		Default Value:	0Ch
		Format:	Opcode
	Render Target Write message		
13	Per-Sample PS Enable		
	Format:	Enable	
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.		
	When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.		
	When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).		
	When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.		
12	Last Render Target Select		
	Format:	Enable	
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select		
	Format:	MDC_RT_SGS	
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype		
	Default Value:	4h	
	Format:	Opcode	
	SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].		

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

	7:0	Binding Table Index
		Format: MDC_BTS
		Specifies the Binding Table Index for the message

Half Precision SIMD16 Render Target Write MSD

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 1h Format: Opcode Half precision data message
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. Programming Notes This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	Message Type

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

		Default Value:	0Ch
		Format:	Opcode
	Render Target Write message		
13	Per-Sample PS Enable		
	Format:	Enable	
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.		
	When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_CO as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.		
	When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).		
	When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.		
12	Last Render Target Select		
	Format:	Enable	
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select		
	Format:	MDC_RT_SGS	
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype		
	Default Value:	0h	
	Format:	Opcode	
	SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].		

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

	7:0	Binding Table Index		
		<table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%; text-align: center;">MDC_BT</td> </tr> </table>	Format:	MDC_BT
Format:	MDC_BT			
		Specifies the Binding Table Index for the message		

Halt

halt - Halt			
Source:	Eulsa		
Length Bias:	4		
<p>The halt instruction temporarily suspends execution for all enabled compute channels. Upon execution, the enabled channels are sent to the instruction at (IP + UIP), if all channels are enabled at HALT, jump to the instruction at (IP + JIP). If the halt instruction is not inside any conditional code block, the values of JIP and UIP should be the same. If the halt instruction is inside a conditional code block, the UIP should be the end of the program and the JIP should be the end of the inner most conditional code block. The UIP must point to a HALT Instruction. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.</p> <p>The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate and dst must be null.</p>			
Format:	[(pred)] halt (exec_size) JIP UIP		
Syntax			
[(pred)] halt (exec_size) imm32 imm32			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn.channel[n]) { PcIP[n] = IP + UIP; else { PcIP[n] = IP + 1; } } if (PcIP != (IP + 1)) { // for all channels Jump(IP + JIP); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
	Format:	MBZ	
	127:96	JIP	
		Exists If:	([Src0.IsImm]==true)
		Format:	S31
The byte-aligned jump distance if a jump is taken for the channel.			
95:80	Reserved		
	Exists If:	([Src0.IsImm]==false)	
Format:	MBZ		

halt - Halt

95:64	Reserved	
	Exists If:	$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == false)$
	Format:	MBZ
95:64	UIP	
	Exists If:	$([Src0.IsImm] == true) \text{ AND } ([Src1.IsImm] == true)$
	Format:	S31
	The byte aligned jump distance if a jump is taken for the instruction.	
79:66	Src0.Operand	
	Exists If:	$([Src0.IsImm] == false)$
	Format:	DirectOperand
65:64	Reserved	
	Exists If:	$([Src0.IsImm] == false)$
	Format:	MBZ
63:50	Dst.Operand	
	Format:	DirectOperand
49:48	Reserved	
	Format:	MBZ
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value	
	Value	Name
	0	false
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value	
	Value	Name
	0	false
	1	true
45:34	Reserved	
	Format:	MBZ
33	BranchCtrl	
	This field is used by <i>goto</i> , <i>if</i> , and <i>andelsei</i> nstructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	

halt - Halt

Value	Name	Description
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
Format:		MBZ
<p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p>		
Value	Name	Description
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
<p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p>		
Value	Name	Description
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl	
Format:		PredCtrl
<p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		
23	FlagRegNum[0]	
<p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	FlagSubRegNum	
<p>This field specifies the sub-register number for a flag register operand. There are two sub-</p>		

halt - Halt			
	<p>registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

HCP_BSD_OBJECT										
		<table border="1"> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>Specifies the length in bytes of the bitstream data for the current slice. It includes the first byte of the slice and the last non-zero byte of the in the slice. Specifically, the zero-padding bytes(if present) and the next start-code are excluded.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[0,268435455]</td> <td>Data_Length_with_28_bits_only</td> <td>Valid range is only from 0 to 268435455, which is corresponding to lower 28 bits. This restriction is for old project which only use 28 bits data length.</td> </tr> </tbody> </table>	Format:	U32	Value	Name	Description	[0,268435455]	Data_Length_with_28_bits_only	Valid range is only from 0 to 268435455, which is corresponding to lower 28 bits. This restriction is for old project which only use 28 bits data length.
Format:	U32									
Value	Name	Description								
[0,268435455]	Data_Length_with_28_bits_only	Valid range is only from 0 to 268435455, which is corresponding to lower 28 bits. This restriction is for old project which only use 28 bits data length.								
2	31:29	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
28:0	<p>Indirect Data Start Address</p> <table border="1"> <tr> <td>Format:</td> <td>U29</td> </tr> </table> <p>Specifies the byte-aligned graphics memory starting address of the slice bit stream relative to the BSD Indirect Object Base Address.</p>	Format:	U29							
Format:	U29									

HCP_FQM_STATE

HCP_FQM_STATE																																					
Source:	VideoCS																																				
Length Bias:	2																																				
<p>The HCP_FQM_STATE command loads the custom HEVC quantization tables into local RAM and may be issued up to 8 times: 4 scaling list per intra and inter.</p>																																					
<p>Driver is responsible for performing the Scaling List division. So, save the division HW cost in HW. The 1/x value is provided in 16-bit fixed-point precision as $((1 \ll 17) / QM + 1) \gg 1$.</p>																																					
<p>Note: FQM is computed as $(2^{16} / QM)$. If $QM=1$, FQM=all 1's.</p>																																					
<p>To simplify the design, only a limited number of scaling lists are provided at the PAK interface: default two SizeID0 and two SizeID123 (one set for inter and the other set for intra), and the encoder only allows custom entries for these four matrices. The DC value of SizeID2 and SizeID3 will be provided.</p>																																					
<p>When the scaling_list_enable_flag is set to disable, the scaling matrix is still sent to the PAK, and with all entries programmed to the same value of 16.</p>																																					
<p>This is a picture level state command and is issued in encoding processes only.</p>																																					
<p>DWords 2-33 form a table for the DCT coefficients, 2 16-bit coefficients/DWord.</p> <ul style="list-style-type: none"> • Size 4x4 for SizeID0, DWords 2-9. • Size 8x8 for SizeID1/2/3, DWords 2-33. 																																					
<p>SizeID 0 (Table 4-13)</p> <table border="1"> <thead> <tr> <th></th> <th style="text-align: center;">4x4</th> <th style="text-align: center;">[31:16]</th> <th style="text-align: center;">[15:0]</th> </tr> </thead> <tbody> <tr> <td>DWord 2</td> <td>AC(0,1)</td> <td colspan="2">DC</td> </tr> <tr> <td>DWord 3</td> <td>AC(0,3)</td> <td colspan="2">AC(0,2)</td> </tr> <tr> <td>DWord 4</td> <td>AC(1,1)</td> <td colspan="2">AC(1,0)</td> </tr> <tr> <td>DWord 5</td> <td>AC(1,3)</td> <td colspan="2">AC(1,2)</td> </tr> <tr> <td>DWord 6</td> <td>AC(2,1)</td> <td colspan="2">AC(2,0)</td> </tr> <tr> <td>DWord 7</td> <td>AC(2,3)</td> <td colspan="2">AC(2,2)</td> </tr> <tr> <td>DWord 8</td> <td>AC(3,1)</td> <td colspan="2">AC(3,0)</td> </tr> <tr> <td>DWord 9</td> <td>AC(3,3)</td> <td colspan="2">AC(3,2)</td> </tr> </tbody> </table>			4x4	[31:16]	[15:0]	DWord 2	AC(0,1)	DC		DWord 3	AC(0,3)	AC(0,2)		DWord 4	AC(1,1)	AC(1,0)		DWord 5	AC(1,3)	AC(1,2)		DWord 6	AC(2,1)	AC(2,0)		DWord 7	AC(2,3)	AC(2,2)		DWord 8	AC(3,1)	AC(3,0)		DWord 9	AC(3,3)	AC(3,2)	
	4x4	[31:16]	[15:0]																																		
DWord 2	AC(0,1)	DC																																			
DWord 3	AC(0,3)	AC(0,2)																																			
DWord 4	AC(1,1)	AC(1,0)																																			
DWord 5	AC(1,3)	AC(1,2)																																			
DWord 6	AC(2,1)	AC(2,0)																																			
DWord 7	AC(2,3)	AC(2,2)																																			
DWord 8	AC(3,1)	AC(3,0)																																			
DWord 9	AC(3,3)	AC(3,2)																																			
<p>SizeID 1, 2, 3 (Table 4-14)</p> <table border="1"> <thead> <tr> <th></th> <th style="text-align: center;">8x8</th> <th style="text-align: center;">[31:16]</th> <th style="text-align: center;">[15:0]</th> </tr> </thead> <tbody> <tr> <td>DWord 2</td> <td>AC(0,1)</td> <td colspan="2">DC</td> </tr> <tr> <td>DWord 3</td> <td>AC(0,3)</td> <td colspan="2">AC(0,2)</td> </tr> <tr> <td>DWord 4</td> <td>AC(0,5)</td> <td colspan="2">AC(0,4)</td> </tr> <tr> <td>DWord 5</td> <td>AC(0,7)</td> <td colspan="2">AC(0,6)</td> </tr> <tr> <td>DWord 6</td> <td>AC(1,1)</td> <td colspan="2">AC(1,0)</td> </tr> <tr> <td>DWord 7</td> <td>AC(1,3)</td> <td colspan="2">AC(1,2)</td> </tr> </tbody> </table>			8x8	[31:16]	[15:0]	DWord 2	AC(0,1)	DC		DWord 3	AC(0,3)	AC(0,2)		DWord 4	AC(0,5)	AC(0,4)		DWord 5	AC(0,7)	AC(0,6)		DWord 6	AC(1,1)	AC(1,0)		DWord 7	AC(1,3)	AC(1,2)									
	8x8	[31:16]	[15:0]																																		
DWord 2	AC(0,1)	DC																																			
DWord 3	AC(0,3)	AC(0,2)																																			
DWord 4	AC(0,5)	AC(0,4)																																			
DWord 5	AC(0,7)	AC(0,6)																																			
DWord 6	AC(1,1)	AC(1,0)																																			
DWord 7	AC(1,3)	AC(1,2)																																			

HCP_FQM_STATE

DWord 8	AC(1,5)	AC(1,4)
DWord 9	AC(1,7)	AC(1,6)
...		
DWord 30	AC(7,1)	AC(7,0)
DWord 31	AC(7,3)	AC(7,2)
DWord 32	AC(7,5)	AC(7,4)
DWord 33	AC(7,7)	AC(7,6)

DWord	Bit	Description					
0	31:29	Command Type <table border="1" style="width: 100%;"> <tr> <td>Default Value:</td> <td>3h PARALLEL_VIDEO_PIPE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	3h PARALLEL_VIDEO_PIPE	Format:	OpCode	
		Default Value:	3h PARALLEL_VIDEO_PIPE				
		Format:	OpCode				
	28:27	Pipeline Type <table border="1" style="width: 100%;"> <tr> <td>Default Value:</td> <td>2h</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	2h	Format:	OpCode	
		Default Value:	2h				
	Format:	OpCode					
	26:23	Media Instruction Opcode <table border="1" style="width: 100%;"> <tr> <td>Default Value:</td> <td>7h Codec/Engine Name</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table> Codec/Engine Name = HCP = 7h	Default Value:	7h Codec/Engine Name	Format:	OpCode	
Default Value:		7h Codec/Engine Name					
Format:	OpCode						
22:16	Media Instruction Command <table border="1" style="width: 100%;"> <tr> <td>Default Value:</td> <td>5h HCP_FQM_STATE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	5h HCP_FQM_STATE	Format:	OpCode		
	Default Value:	5h HCP_FQM_STATE					
Format:	OpCode						
15:12	Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ					
11:0	Dword Length <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> (Excludes Dwords 0, 1). <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="background-color: #e1eef6;">Value</th> <th style="background-color: #e1eef6;">Name</th> </tr> </thead> <tbody> <tr> <td>20h</td> <td></td> </tr> </tbody> </table>	Format:	=n	Value	Name	20h	
	Format:	=n					
	Value	Name					
20h							
1	31:16	FQM DC Value: (1/DC): <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U16</td> </tr> </table> Specifies DC value of the scaling list for 16x16 (SizeID=2) or 32x32 (SizeID=3). DC Value = scaling_list_dc_coef_minus8 + 8. Driver will do the division.	Format:	U16			
		Format:	U16				
		15:5	Reserved				

HCP_FQM_STATE			
		Format: MBZ	
	4:3	Color Component	
		Format: U2	
		Luma and Chroma's share the same scaling list and DC value for the same SizeID.	
		Value	Name
		0	Luma
	1	Chroma Cb	
	2	Chroma Cr	
	3	Reserved	
	2:1	SizeID	
		Format: U2	
Value		Name	
0		SizeID 0 4x4	
1		SizeID 1, 2, 3 (8x8, 16x16, 32x32)	
0	Intra/Inter		
	Format: U1		
	This field specifies the quant matrix intra or inter type.		
	Value	Name	
	0	Intra	
1	Inter		
2..33	1023:0	QuantizerMatrix	

HCP_IND_OBJ_BASE_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE				
Source:	VideoCS			
Length Bias:	2			
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The HCP_IND_OBJ_BASE_ADDR_STATE command is used to define the indirect object base address of the stream in graphics memory. This is a frame level command. (Is it frame or picture level?) This is a picture level state command and is issued in both encoding and decoding processes.</p>				
Compressed Header Format				
Fields	Bits			
Bin	0	HuC/Kernel Binarized Syntax		
Probability select	1	0 -> indicates probability 128 1 -> indicates probability 256		
	Repeat to pack a Cacheline			
Partition1 and TileSize record				
Fields	Bits			
Tile Size	31:0	Partition1 Size is 16-bit value, Tile Size is 32-bit value		
AddressOffset	63:32	Cacheline Address Offset to be Modified		
Offset	69:64	Byte offset to be Modified		
16-bit vs 32-bit update	70	0: Update 16-bit; 1: Update 32-bit		
Reserved	511:71			
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27		Pipeline Type	
			Default Value:	2h
			Format:	OpCode
	26:23		Media Instruction Opcode	
			Default Value:	7h Codec/Engine Name
			Format:	OpCode
			Codec/Engine Name = HCP = 7h	
	22:16		Media Instruction Command	
			Default Value:	3h HCP_IND_OBJ_BASE_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE						
		Format: OpCode				
	15:12	Reserved Format: MBZ				
	11:0	Dword Length Format: =n (Excludes Dwords 0, 1). <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>Ch</td> <td></td> </tr> </tbody> </table>	Value	Name	Ch	
Value	Name					
Ch						
1..2	63:0	HCP Indirect Bitstream Object Base Address Format: SplitBaseAddress4KByteAligned Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the HCP_BSD_OBJECT command for fetching (reading) the compressed Slice Data.				
3	31:0	HCP Indirect Bitstream Object Memory Address Attributes Format: MemoryAddressAttributes				
4..5	63:0	HCP Indirect Bitstream Object Access Upper Bound Format: SplitBaseAddress4KByteAligned Decoder only. This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the HCP_BSD_OBJECT command for the slice bit stream. Indirect data accessed at this address or greater will cause the HCP to stop issuing requests to the GAC and the BSP VLD will then only receive zeros until a slice done is received. Setting this field to 0 will cause this range to be ignored by the HCP.				
6..7	63:0	HCP Indirect CU Object Base Address Format: BaseAddress4KByteAligned Encoder only. Specifies the 4K-byte aligned data buffer base address for the read-only indirect data object for fetching (reading) per CU data during the encoding process.				
8	31:0	HCP Indirect CU Object Object Memory Address Attributes Format: MemoryAddressAttributes				
9..10	63:0	HCP PAK-BSE Object Base Address Format: BaseAddress4KByteAligned Encoder only. Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the compressed bitstream.				
11	31:0	HCP PAK-BSE Object Address Memory Address Attributes				

HCP_IND_OBJ_BASE_ADDR_STATE									
	<table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> <tr> <td colspan="2">Encoder only.</td> </tr> </table>	Format:	MemoryAddressAttributes	Encoder only.					
Format:	MemoryAddressAttributes								
Encoder only.									
12..13	<table border="1"> <tr> <td>63:0</td> <td>HCP PAK-BSE Object Access Upper Bound</td> </tr> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> <tr> <td colspan="2">Encoder only.</td> </tr> <tr> <td colspan="2">This field specifies the 4K-byte aligned maximum memory address access by the HCP_PAK_OBJECT command for writing out the slice bit stream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. This address must be greater than the HCP PAK-BSE Object Base Address state.</td> </tr> </table>	63:0	HCP PAK-BSE Object Access Upper Bound	Format:	SplitBaseAddress4KByteAligned	Encoder only.		This field specifies the 4K-byte aligned maximum memory address access by the HCP_PAK_OBJECT command for writing out the slice bit stream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. This address must be greater than the HCP PAK-BSE Object Base Address state.	
63:0	HCP PAK-BSE Object Access Upper Bound								
Format:	SplitBaseAddress4KByteAligned								
Encoder only.									
This field specifies the 4K-byte aligned maximum memory address access by the HCP_PAK_OBJECT command for writing out the slice bit stream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. This address must be greater than the HCP PAK-BSE Object Base Address state.									
14..15	<table border="1"> <tr> <td>63:0</td> <td>HCP VP9 PAK Compressed Header Syntax Streamin- Base Address</td> </tr> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>BaseAddress4KByteAligned</td> </tr> <tr> <td colspan="2">Specifies the 4K-byte aligned data buffer base address for the read-only Probability counters fetching during the encoding process..</td> </tr> </table>	63:0	HCP VP9 PAK Compressed Header Syntax Streamin- Base Address	Exists If:	//Encoder Only	Format:	BaseAddress4KByteAligned	Specifies the 4K-byte aligned data buffer base address for the read-only Probability counters fetching during the encoding process..	
63:0	HCP VP9 PAK Compressed Header Syntax Streamin- Base Address								
Exists If:	//Encoder Only								
Format:	BaseAddress4KByteAligned								
Specifies the 4K-byte aligned data buffer base address for the read-only Probability counters fetching during the encoding process..									
16	<table border="1"> <tr> <td>31:0</td> <td>HCP VP9 PAK Compressed Header Syntax StreamIn Memory Address Attributes</td> </tr> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	31:0	HCP VP9 PAK Compressed Header Syntax StreamIn Memory Address Attributes	Format:	MemoryAddressAttributes				
31:0	HCP VP9 PAK Compressed Header Syntax StreamIn Memory Address Attributes								
Format:	MemoryAddressAttributes								
17..18	<table border="1"> <tr> <td>63:0</td> <td>HCP VP9 PAK Probability Counter StreamOut- Base Address</td> </tr> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>BaseAddress4KByteAligned</td> </tr> <tr> <td colspan="2">Specifies the 4K-byte aligned data buffer base address for the write-only Probability counters fetching during the encoding process..</td> </tr> </table>	63:0	HCP VP9 PAK Probability Counter StreamOut- Base Address	Exists If:	//Encoder Only	Format:	BaseAddress4KByteAligned	Specifies the 4K-byte aligned data buffer base address for the write-only Probability counters fetching during the encoding process..	
63:0	HCP VP9 PAK Probability Counter StreamOut- Base Address								
Exists If:	//Encoder Only								
Format:	BaseAddress4KByteAligned								
Specifies the 4K-byte aligned data buffer base address for the write-only Probability counters fetching during the encoding process..									
19	<table border="1"> <tr> <td>31:0</td> <td>HCP VP9 PAK Probability Counter StreamOut Memory Address Attributes</td> </tr> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	31:0	HCP VP9 PAK Probability Counter StreamOut Memory Address Attributes	Exists If:	//Encoder Only	Format:	MemoryAddressAttributes		
31:0	HCP VP9 PAK Probability Counter StreamOut Memory Address Attributes								
Exists If:	//Encoder Only								
Format:	MemoryAddressAttributes								
20..21	<table border="1"> <tr> <td>63:0</td> <td>HCP VP9 PAK Probability Deltas StreamIn- Base Address</td> </tr> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>BaseAddress4KByteAligned</td> </tr> <tr> <td colspan="2">Specifies the 4K-byte aligned data buffer base address for the read-only Probability differences during the encoding process.</td> </tr> </table>	63:0	HCP VP9 PAK Probability Deltas StreamIn- Base Address	Exists If:	//Encoder Only	Format:	BaseAddress4KByteAligned	Specifies the 4K-byte aligned data buffer base address for the read-only Probability differences during the encoding process.	
63:0	HCP VP9 PAK Probability Deltas StreamIn- Base Address								
Exists If:	//Encoder Only								
Format:	BaseAddress4KByteAligned								
Specifies the 4K-byte aligned data buffer base address for the read-only Probability differences during the encoding process.									
22	<table border="1"> <tr> <td>31:0</td> <td>HCP VP9 PAK Probability Deltas StreamIn Memory Address Attributes</td> </tr> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	31:0	HCP VP9 PAK Probability Deltas StreamIn Memory Address Attributes	Exists If:	//Encoder Only	Format:	MemoryAddressAttributes		
31:0	HCP VP9 PAK Probability Deltas StreamIn Memory Address Attributes								
Exists If:	//Encoder Only								
Format:	MemoryAddressAttributes								
23..24	<table border="1"> <tr> <td>63:0</td> <td>HCP VP9 PAK Tile Record StreamOut- Base Address</td> </tr> <tr> <td>Format:</td> <td>BaseAddress4KByteAligned</td> </tr> </table>	63:0	HCP VP9 PAK Tile Record StreamOut- Base Address	Format:	BaseAddress4KByteAligned				
63:0	HCP VP9 PAK Tile Record StreamOut- Base Address								
Format:	BaseAddress4KByteAligned								

HCP_IND_OBJ_BASE_ADDR_STATE						
		Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the Tile Record.				
25	31:0	HCP VP9 PAK Tile Record StreamOut Memory Address Attributes <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Exists If:	//Encoder Only	Format:	MemoryAddressAttributes
Exists If:	//Encoder Only					
Format:	MemoryAddressAttributes					
26..27	63:0	HCP VP9 PAK CU Level Statistic StreamOut- Base Address <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>BaseAddress4KByteAligned</td> </tr> </table> <p>Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the CU Record.</p>	Exists If:	//Encoder Only	Format:	BaseAddress4KByteAligned
Exists If:	//Encoder Only					
Format:	BaseAddress4KByteAligned					
28	31:0	HCP VP9 PAK CU Level Statistic StreamOut Memory Address Attributes <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Exists If:	//Encoder Only	Format:	MemoryAddressAttributes
Exists If:	//Encoder Only					
Format:	MemoryAddressAttributes					

HCP_PAK_INSERT_OBJECT

HCP_PAK_INSERT_OBJECT		
Source:	VideoCS	
Length Bias:	2	
<p>It is an encoder only command, operating at bitstream level, before and after SliceData compressed bitstream. It is setup by the header and tail present flags in the Slice State command. If these flags are set and no subsequent PAK_INSERT_OBJECT commands are issued, the pipeline will hang.</p>		
<p>The HCP_PAK_INSERT_OBJECT command supports both inline and indirect data payload, but only one can be active at any time. It is issued to insert a chunk of bits (payload) into the current compressed bitstream output buffer (specified in the HCP_PAK-BSE Object Base Address field of the HCP_IND_OBJ_BASE_ADDR_STATE command) starting at its current write pointer bit position. Hardware will keep track of this write pointer's byte position and the associated next bit insertion position index.</p>		
<p>It is a variable length command when the payload (data to be inserted) is presented as inline data within the command itself. The inline payload is a multiple of 32-bit (1 DW), as the data bus to the compressed bitstream output buffer is 32-bit wide.</p>		
<p>The payload data is required to be byte aligned on the left (first transmitted bit order) and may or may not be byte aligned on the right (last transmitted bits). The command will specify the bit offset of the last valid DW. Note that : Stitch Command is used if the beginning position of data is in bit position. When PAK Insert Command is used the beginning position must be in byte position.</p>		
<p>Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid bitstream.</p>		
<p>Internally, HCP hardware will keep track of the very last two bytes' (the very last byte can be a partial byte) values of the previous insertion. It is required that the next Insertion Object Command or the next PAK Object Command to perform the start code emulation sequence check and prevention 0x03 byte insertion with this end condition of the previous insertion.</p>		
<p>The payload data may have already been processed for start code emulation byte insertion, except the possibility of the last 2 bytes plus the very last partial byte (if any). Hence, when hardware performing the concatenation of multiple consecutive insertion commands, or concatenation of an insertion command and a PAK object command, it must check and perform the necessary start code emulation byte insert at the junction.</p>		
<p>Data to be inserted can be a valid NAL units or a partial NAL unit. It can be any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current Slice - SPS NAL, PPS NAL, SEI NAL and Other Non-Slice NAL, Leading_Zero_8_bits (as many bytes as there is), Start Code , Slice Header. Any encoded syntax elements bit data after the encoded Slice Data (PAK Object Command) of the current Slice and prior to the next encoded Slice Data of the next Slice or prior to the end of the bitstream, whichever comes first Cabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is).</p>		
<p>Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by SLICE STATE Command) determines the number of CABAC_ZERO_WORD to be inserted to the end of the current NAL, based on the minimum byte size of a NAL and the actual bin count of the encoded Slice. Since prior to the CABAC_ZERO_WORD insertion, the RBSP or EBSP is already byte-aligned, so each CABAC_ZERO_WORD insertion is actually a 3-byte sequence 0x00 00 03.</p>		
<p>Context switch interrupt is not supported by this command.</p>		
DWord	Bit	Description

HCP_PAK_INSERT_OBJECT																
0	31:29	Command Type <table border="1"> <tr> <td>Default Value:</td> <td>3h PARALLEL_VIDEO_PIPE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	3h PARALLEL_VIDEO_PIPE	Format:	OpCode										
	Default Value:	3h PARALLEL_VIDEO_PIPE														
	Format:	OpCode														
	28:27	Pipeline Type <table border="1"> <tr> <td>Default Value:</td> <td>2h</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	2h	Format:	OpCode										
	Default Value:	2h														
	Format:	OpCode														
	26:23	Media Instruction Opcode <table border="1"> <tr> <td>Default Value:</td> <td>7h Codec/Engine Name</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table> <p>Codec/Engine Name = HCP = 7h</p>	Default Value:	7h Codec/Engine Name	Format:	OpCode										
Default Value:	7h Codec/Engine Name															
Format:	OpCode															
22:16	Media Instruction Command <table border="1"> <tr> <td>Default Value:</td> <td>22h HCP_PAK_INSERT_OBJECT</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	22h HCP_PAK_INSERT_OBJECT	Format:	OpCode											
Default Value:	22h HCP_PAK_INSERT_OBJECT															
Format:	OpCode															
15:12	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ															
11:0	Dword Length <table border="1"> <tr> <td>Default Value:</td> <td>[1h, FFFh] DWORD_COUNT_n</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>(Excludes Dwords 0, 1) = Total Length - 2, so DWord Length = X, where X is in the size of the payload in DWs 2..n which has the range of [1,4095]</p>	Default Value:	[1h, FFFh] DWORD_COUNT_n	Format:	=n											
Default Value:	[1h, FFFh] DWORD_COUNT_n															
Format:	=n															
1	31	Indirect Payload Enable <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Only one of these two payload modes can be active at any time. When Slice Size Conformance is enable the Payload(header) must be inline only so this bit set to MBZ.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>inline payload is used</td> <td></td> </tr> <tr> <td>1</td> <td>indirect payload is used</td> <td></td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>In VP9 scalability mode, tail should be inserted in inline mode only so this bit Must Be zero.</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	Description	0	inline payload is used		1	indirect payload is used		Programming Notes		In VP9 scalability mode, tail should be inserted in inline mode only so this bit Must Be zero.
	Format:	Enable														
	Value	Name	Description													
	0	inline payload is used														
1	indirect payload is used															
Programming Notes																
In VP9 scalability mode, tail should be inserted in inline mode only so this bit Must Be zero.																
30:18	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ															
17:16	DataByteOffset - SrcDataStartingByteOffset[1:0] <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <p>Source Data Starting Byte Position within the very first inline DW.</p>	Format:	U2													
Format:	U2															

HCP_PAK_INSERT_OBJECT

15	HeaderLengthExcludeFrmSize										
	Format:	U1									
	<p>In case this flag is on, bits are NOT accumulated during current access unit coding neither for Cabac Zero Word insertion bits counting or for output in MMIO register HCP_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER.</p> <p>When using HeaderLengthExcludeFrmSize for header insertion, the software needs to make sure that data comes already with inserted start code emulation bytes. SW shouldn't set EmulationFlag bit (Bit 3 of DWORD1 of HCP_PAK_INSERT_OBJECT).</p>										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>All bits accumulated</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Bits during current call are not accumulated</td> </tr> </tbody> </table>		Value	Description	0	All bits accumulated	1	Bits during current call are not accumulated			
Value	Description										
0	All bits accumulated										
1	Bits during current call are not accumulated										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 60%;">Name</th> <th style="width: 25%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>All bits accumulated</td> <td></td> </tr> <tr> <td style="text-align: center;">1</td> <td>Bits during current call are not accumulated</td> <td></td> </tr> </tbody> </table>		Value	Name	Description	0	All bits accumulated		1	Bits during current call are not accumulated	
Value	Name	Description									
0	All bits accumulated										
1	Bits during current call are not accumulated										
	Programming Notes										
	Must be set to 0 for JPEG encoder.										
14	Slice Header Indicator										
	Format:	U1									
	<p>This bit indicates if the insert object is a slice header. In the VDEnc mode, PAK only gets this command at the beginning of the frame for slice position X=0, Y=0. It internally generates the header that needs to be inserted per slice. For VDEnc mode, this bit should always be set.</p>										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>SLICE_HEADER</td> <td>Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.</td> </tr> <tr> <td style="text-align: center;">0</td> <td>LEGACY</td> <td>Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.</td> </tr> </tbody> </table>		Value	Name	Description	1	SLICE_HEADER	Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.	0	LEGACY	Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.
Value	Name	Description									
1	SLICE_HEADER	Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.									
0	LEGACY	Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.									
	Programming Notes										
	<p>In VDENC mode, we support only Slice layer without partitioning RBSP syntax. The payload for PAK_INS_OBJ should contain only start code for Slice header followed by NAL_type and slice header (slice_header() in HEVC spec).</p> <p>The payload for PAK_INS_OBJ shouldn't contain CABAC Byte alignment bits. HW adds these alignment bits which are part of slice_data.</p> <p>Example PAK_INS_OBJ payload : 00 00 01 <NAL_type> <slice_header_Byte0> ..<slice_header_Byte LAST></p> <p>Any zero_bytes that are added before slice header can be inserted by any preceding general PAK_INS_OBJ.</p>										
13:8	DataBitsInLastDW - SrCDataEndingBitInclusion[5:0]										

HCP_PAK_INSERT_OBJECT

		Format:	U6
		<p>Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first. For example, SrCDataEndingBitInclusion = 9, bit 7:0 and bit 15 are included as valid header data.</p> <p>For VP9: The Driver has to give byte aligned Data for the last inline DW. (the driver pads Zeros to next byte boundary to the original header if it was not byte aligned on the last inline DW).</p>	
		Value	Name
		[1,32]	
7:4	SkipEmulByteCnt - Skip Emulation Byte Count		
		Format:	U4
		<p>Skip emulation check for number of starting bytes It can be programmed from 0 to 15 bytes. For example, to skip the start code that has already prefixed in the bitstream.</p> <p>Only valid for HEVC and reserved for VP9.</p>	
3	EmulationFlag - EmulationByteBitsInsertEnable		
		Format:	U1
		Only valid for HEVC and reserved for VP9.	
	Value	Name	Description
	0		When this bit is set to 0, HW will disable the insertion of emulation bytes into the bitstream.
	1		When this bit is set to 1, HW will enable the insertion of emulation bytes into the bitstream.
2	Las theaderFlag - LastSrcHeaderDataInsertCommandFlag		
		Format:	U1
		<p>To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series. In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command.</p>	
1	EndOfSliceFlag - LastDstDataInsertCommandFlag		
		Format:	U1
		No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory.	
0	Reserved		
		Format:	MBZ
2..n	127:0	Indirect Payload	
		Exists If:	((Indirect Payload Enable)== 1)
		Format:	HCP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD

HCP_PAK_INSERT_OBJECT					
31:0	<p>Inline Payload</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Exists If:</td> <td>(([Indirect Payload Enable] == 0)</td> </tr> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>Actual Data (inline) to be inserted to the output bitstream buffer.</p>	Exists If:	(([Indirect Payload Enable] == 0)	Format:	U32
Exists If:	(([Indirect Payload Enable] == 0)				
Format:	U32				

HCP_PAK_OBJECT

HCP_PAK_OBJECT			
Source:		VideoCS	
Length Bias:		2	
This is a per-LCU command for encoder only.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = HCP = 7h	
22:16	Media Instruction Command		
	Default Value:	21h HCP_PAK_OBJECT	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	1h		
3h			
1	31	LastCtbOfSlice Flag	
		Format:	Boolean
		Value	Name
		0	False
	1	True	
30	LastCtbOfTile Flag		
Format:	U1		
Indicates last LCU or SB of a Tile			

HCP_PAK_OBJECT				
	29:24	CU_count_minus1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U6</td></tr></table> Number of CUs in the current LCU = CU_count_minus1 + 1. Minimum, there must be 1 CU in a LCU.		U6
	U6			
	23:21	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">MBZ</td></tr></table>		MBZ
	MBZ			
	20:0	split_coding_unit_flag[x0][y0] Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">Split_coding_unit_flags</td></tr></table> If CU size=64x64 and DQP!=0, split should happen at least once		Split_coding_unit_flags
	Split_coding_unit_flags			
2	31:16	Current LCU Y Addr Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U16</td></tr></table>		U16
		U16		
15:0	Current LCU X Addr Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U16</td></tr></table>		U16	
	U16			
3	31:0	Estimated LCU Size in Bytes Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U32</td></tr></table> This parameter indicates the estimated LCU size in bytes. This is generated in the RD Estimator by adding the Coeff bytes of the LCU to the CU costs (pre-lambda input values).		U32
	U32			
4	31:18	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">MBZ</td></tr></table>		MBZ
		MBZ		
	17	Reserved		
	16	LcuForceZeroCoeff (Time Budget Overflow Occurred) Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">Enable</td></tr></table> The bitstream packer pipeline will force all the coefficients to zero. Also, VDenc indicates PAK that time budget overflow has occurred.		Enable
		Enable		
	15:12	SSE ClassID 32x32_3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U4</td></tr></table> This parameter indicates the SSE classID for the 32x32 block3 of the current LCU. Valid values: 0-8		U4
		U4		
	11:8	SSE ClassID 32x32_2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U4</td></tr></table> This parameter indicates the SSE classID for the 32x32 block2 of the current LCU. Valid values: 0-8		U4
	U4			
7:4	SSE ClassID 32x32_1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U4</td></tr></table> This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8		U4	
	U4			
3:0	SSE ClassID 32x32_0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">U4</td></tr></table>		U4	
	U4			

HCP_PAK_OBJECT		
-----------------------	--	--

		This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8
--	--	--

HCP_PALETTE_INITIALIZER_STATE

HCP_PALETTE_INITIALIZER_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p>			
<p>The HCP_PALETTE_INITIALIZER_STATE command loads in the SCC Palette Initializer Table to the HW. Decoder only command.</p>			
<p>Dword#2 - 193form a fixed size table for the Palette Initializer Table. Max Palette Initializer Table is 128entries. Each entry has 3 components (Y, Cb and Cr) for a color. Each component is 16-bits, even though currently only support up to 10-bit SCC extension. The upper (higher bits) 6 bits are set to zero - that is Least Significant Bit alignment. Each entry of the Palette Initializer Table will consume 1.5 Dwords. Every two entries will consume 2 Dwords. Hence, total requires 96 Dwords. Dword#2 Bit 31 Cb#0 15:0 Luma#0 15:0 Bit 0 Dword#3 Bit 31 Luma#115:0 Cr#015:0 Bit 0 Dword#4 Bit 31 Cr#115:0 Cb#115:0 Bit 0 Dword#2 corresponds to the entry# 0 of the Palette Initializer Table. Dword#193correspondsto the entry# 127of the Palette Initializer Table.</p>			
Programming Notes			
<p>Palette Initialization needs to happen at the beginning of each frame/tiles or start of each independent slice. Palette initialization is not needed at the start of dependent slices (except the start of a new tiles since each tile needs to re-initialize the palette list) and the palette list is inherited from previous slice. The following is the programming restriction: (1) Palette Initialization command must be programmed in palette mode at the beginning of each frame and tiles (regardless if the slice is independent/dependent) and also the start of each independent slices. (2) Palette Initialization command must not be programmed for dependent slices except the dependent slices are start of tiles (first slice in frame must be independent slice).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	Opcode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	Opcode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
		Format:	OpCode
Codec/Engine Name = HCP = 7h			

HCP_PALETTE_INITIALIZER_STATE							
	22:16	Media Instruction Command <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Default Value:</td> <td>9h HCP_PALETTE_INITIALIZER_STATE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	9h HCP_PALETTE_INITIALIZER_STATE	Format:	OpCode	
	Default Value:	9h HCP_PALETTE_INITIALIZER_STATE					
	Format:	OpCode					
	15:12	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ						
11:0	Dword Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>=n</td> </tr> </table> (Excludes Dwords 0, 1) <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td>C0h</td> <td></td> </tr> </tbody> </table>	Format:	=n	Value	Name	C0h	
Format:	=n						
Value	Name						
C0h							
1	31:8	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
	Format:	MBZ					
7:0	Active Palette Initializer Table Entries The number of entries in the Palette Initializer Table that is valid. The Palette Initializer Table always filled with only valid entries starting from entry 0 onwards, packed and no jump. Max allowed 128entries. This field is set to 0 if there is no active color entry.						
2..97	3071:0	First 64 Color Entries This contains first 64 color entries (0 to 63), each with 16-bit Y, U, V entries. DW2 = [31:0]; DW3 = [63:32], etc for (i from 0 to 63) Palette Initializer Luma Value $i = [(3 * i * 16 + 15) : (3 * i * 16)]$ Palette Initializer Cb Value $i = [((3 * i + 1) * 16 + 15) : ((3 * i + 1) * 16)]$ Palette Initializer CrValue $i = [((3 * i + 2) * 16 + 15) : ((3 * i + 2) * 16)]$					
98..193	3071:0	Second 64 Color Entries This contains second 64 color entries (64 to 127), each with 16-bit Y, U, V entries. DW98 = [31:0]; DW99 = [63:32], etc for (i from 64 to 127) Palette Initializer Luma Value $i = [(3 * (i - 64) * 16 + 15) : (3 * (i - 64) * 16)]$ Palette Initializer Cb Value $i = [((3 * (i - 64) + 1) * 16 + 15) : ((3 * (i - 64) + 1) * 16)]$ Palette Initializer CrValue $i = [((3 * (i - 64) + 2) * 16 + 15) : ((3 * (i - 64) + 2) * 16)]$					

HCP_PIC_STATE

HCP_PIC_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>This is a picture level command and is issued only once per workload for both encoding and decoding processes.</p>			
DWord	Bit	Description	
0	31:29	Command Type Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode	
		28:27	Pipeline Type Default Value: 2h Format: OpCode
			26:23
	22:16	Media Instruction Command Default Value: 10h HCP_PIC_STATE Format: OpCode	
		15:12	Reserved Format: MBZ
	11:0		Dword Length Format: =n (Excludes Dwords 0, 1).
		1	31:27
26:16	FrameHeightInMinCbMinus1 Format: U11 Specifies the height of each decoded picture in units of minimum coding block size.		
	Programming Notes The decoded picture height in units of luma samples equals <ul style="list-style-type: none"> • (FrameHeightInMinCbMinus1 + 1) * 		

HCP_PIC_STATE														
		<ul style="list-style-type: none"> • $(1 \ll (\log_2_{\text{min_coding_block_size_minus3}} + 3))$ • The maximum support picture size is 16384 pixels for both encoder and decoder. <p>The decoded picture height in units of luma samples equals $(\text{FrameHeightInMinCbMinus1} + 1) * (1 <$</p> <p>In HEVC Encoder mode, the following restrictions apply. Note : for a frame width that is not an integer multiple of LCU : Kernel : on last LCU at frames right edge, it must align to CU boundary. This applies to all size of LCU: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. Driver : sets up a LCU aligned (both in X/Y direction) surface.</p>												
	15	<p>PAK Transform Skip Enable If global bit, transform_skip_enabled_flag in image state dw4.22 set to 1 and this bit set to, 1 -> HW will perform transform skip and over write the ENC decision in CU record 0-> HW will not perform transform skip if transform_skip_enabled_flag=0, no transform skip performed in HW or ENC</p> <p style="text-align: center;">Programming Notes</p> <p>Encoder Only</p>												
	14:11	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ													
	10:0	<p>FrameWidthInMinCbMinus1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U11</td> </tr> </table> <p>Specifies the width of each decoded picture in units of minimum coding block size.</p> <p style="text-align: center;">Programming Notes</p> <p>The decoded picture width in units of luma samples equals $(\text{FrameWidthInMinCbMinus1} + 1) * (1 <$</p> <p>The picture width in units of luma samples equals</p> <ul style="list-style-type: none"> • $(\text{FrameWidthInMinCbMinus1} + 1) *$ • $(1 \ll (\log_2_{\text{min_coding_block_size_minus3}} + 3))$ • The maximum support picture size is 16384 pixels for both encoder and decoder. 	Format:	U11										
Format:	U11													
2	31:29	<p>Chroma Subsampling Specify the chroma subsampling of the current bitstream to be decoded or encoded.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Reserved</td> <td>This value is reserved for Monochrome Format which is not supported currently</td> </tr> <tr> <td>1h</td> <td>4:2:0</td> <td>4:2:0 Format</td> </tr> <tr> <td>2h</td> <td>4:2:2</td> <td>4:2:2 Format</td> </tr> </tbody> </table>	Value	Name	Description	0h	Reserved	This value is reserved for Monochrome Format which is not supported currently	1h	4:2:0	4:2:0 Format	2h	4:2:2	4:2:2 Format
Value	Name	Description												
0h	Reserved	This value is reserved for Monochrome Format which is not supported currently												
1h	4:2:0	4:2:0 Format												
2h	4:2:2	4:2:2 Format												

HCP_PIC_STATE										
	3h	4:4:4 4:4:4 Format								
28	<p>chroma_qp_offset_list_enabled_flag 0 cu_chroma_qp_offset_flag is not present (default) 1 cu_chroma_qp_offset_flag is present.</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td>Programming Notes</td> </tr> <tr> <td>Only for 4:4:4 can it be program to non-zero. Decoder Only</td> </tr> </table>		Programming Notes	Only for 4:4:4 can it be program to non-zero. Decoder Only						
Programming Notes										
Only for 4:4:4 can it be program to non-zero. Decoder Only										
27:24	<p>diff_cu_chroma_qp_offset_depth Difference between the luma coding tree block size and the minimum luma coding block size of coding units that convey cu_chroma_qp_offset_flag. $\text{Log2MinCuChromaQpOffsetSize} = \text{CtbLog2SizeY} - \text{diff_cu_chroma_qp_offset_depth}$ 0 to log2_diff_max_min_luma_coding_block_size default = 0. Decoder only feature</p>									
23	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ						
Format:	MBZ									
22:20	<p>chroma_qp_offset_list_len_minus1 It is the index value i of the cb_qp_offset_list[i] and cr_qp_offset_list[i] syntax elements that are present in the PPS i = 0 to 5 6 to 7 are invalid valid. default = 0 Decoder only feature</p>									
19	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ						
Format:	MBZ									
18:16	<p>log2_sao_offset_scale_chroma To scale SAO offset values for chroma samples. 0 to Max(0, BitDepthc10) default = 0 Decoder Only</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0 [Default]</td> </tr> <tr> <td>1</td> <td>1</td> </tr> <tr> <td>2</td> <td>2</td> </tr> </tbody> </table>		Value	Name	0	0 [Default]	1	1	2	2
Value	Name									
0	0 [Default]									
1	1									
2	2									
15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ						
Format:	MBZ									
14:12	<p>log2_sao_offset_scale_luma To scale SAO offset values for luma samples 0 to Max(0, BitDepthC10) Default = 0</p>									

HCP_PIC_STATE															
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>0 [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>1</td> </tr> <tr> <td style="text-align: center;">2</td> <td>2</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Decoder only</td> </tr> </tbody> </table>	Value	Name	0	0 [Default]	1	1	2	2	Programming Notes	Decoder only				
Value	Name														
0	0 [Default]														
1	1														
2	2														
Programming Notes															
Decoder only															
11:10	<p>MaxPCMSize</p> <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <p>Specifies the largest allowed PCM coding block size.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td>Reserved</td> </tr> <tr> <td style="text-align: center;">2</td> <td>32x32</td> </tr> <tr> <td style="text-align: center;">1</td> <td>16x16</td> </tr> <tr> <td style="text-align: center;">0</td> <td>8x8</td> </tr> </tbody> </table>	Format:	U2	Value	Name	3	Reserved	2	32x32	1	16x16	0	8x8		
Format:	U2														
Value	Name														
3	Reserved														
2	32x32														
1	16x16														
0	8x8														
9:8	<p>MinPCMSize</p> <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <p>Specifies the smallest allowed PCM coding block size.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td>Reserved</td> </tr> <tr> <td style="text-align: center;">2</td> <td>32x32</td> </tr> <tr> <td style="text-align: center;">1</td> <td>16x16</td> </tr> <tr> <td style="text-align: center;">0</td> <td>8x8</td> </tr> </tbody> </table>	Format:	U2	Value	Name	3	Reserved	2	32x32	1	16x16	0	8x8		
Format:	U2														
Value	Name														
3	Reserved														
2	32x32														
1	16x16														
0	8x8														
7:6	<p>MaxTUSize</p> <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <p>Specifies the largest allowed transform block size.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td>32x32</td> </tr> <tr> <td style="text-align: center;">2</td> <td>16x16</td> </tr> <tr> <td style="text-align: center;">1</td> <td>8x8</td> </tr> <tr> <td style="text-align: center;">0</td> <td>4x4</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>IN VDENC mode this field should always be set to 3.</td> </tr> </tbody> </table>	Format:	U2	Value	Name	3	32x32	2	16x16	1	8x8	0	4x4	Programming Notes	IN VDENC mode this field should always be set to 3.
Format:	U2														
Value	Name														
3	32x32														
2	16x16														
1	8x8														
0	4x4														
Programming Notes															
IN VDENC mode this field should always be set to 3.															
5:4	<p>MinTUSize</p> <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <p>Specifies the smallest allowed transform block size.</p>	Format:	U2												
Format:	U2														

HCP_PIC_STATE																					
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td>32x32</td> </tr> <tr> <td style="text-align: center;">2</td> <td>16x16</td> </tr> <tr> <td style="text-align: center;">1</td> <td>8x8</td> </tr> <tr> <td style="text-align: center;">0</td> <td>4x4</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>In VDENC mode this bit should be set to 0.</td> </tr> </tbody> </table>	Value	Name	3	32x32	2	16x16	1	8x8	0	4x4	Programming Notes	In VDENC mode this bit should be set to 0.							
Value	Name																				
3	32x32																				
2	16x16																				
1	8x8																				
0	4x4																				
Programming Notes																					
In VDENC mode this bit should be set to 0.																					
	3:2	<p>CtbSize (LCUSize)</p> <table border="1"> <tr> <td>Format:</td> <td style="text-align: center;">U2</td> </tr> </table> <p>Specifies the coding tree block size.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td>64x64</td> <td>In VDENC, mode this should be set to 3.</td> </tr> <tr> <td style="text-align: center;">2</td> <td>32x32</td> <td></td> </tr> <tr> <td style="text-align: center;">1</td> <td>16x16</td> <td>This can only be used when both picture width and height are fewer than or equal to 4222 pixels.</td> </tr> <tr> <td style="text-align: center;">0</td> <td>illegal/reserved</td> <td></td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>LCU is restricted based on the picture size. LCU 16x16 can only be used with picture width and height both fewer than or equal to 4222 pixels.</td> </tr> </tbody> </table>	Format:	U2	Value	Name	Programming Notes	3	64x64	In VDENC, mode this should be set to 3.	2	32x32		1	16x16	This can only be used when both picture width and height are fewer than or equal to 4222 pixels.	0	illegal/reserved		Programming Notes	LCU is restricted based on the picture size. LCU 16x16 can only be used with picture width and height both fewer than or equal to 4222 pixels.
Format:	U2																				
Value	Name	Programming Notes																			
3	64x64	In VDENC, mode this should be set to 3.																			
2	32x32																				
1	16x16	This can only be used when both picture width and height are fewer than or equal to 4222 pixels.																			
0	illegal/reserved																				
Programming Notes																					
LCU is restricted based on the picture size. LCU 16x16 can only be used with picture width and height both fewer than or equal to 4222 pixels.																					
	1:0	<p>MinCUSize</p> <table border="1"> <tr> <td>Format:</td> <td style="text-align: center;">U2</td> </tr> </table> <p>Specifies the smallest coding block size.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td>64x64</td> </tr> <tr> <td style="text-align: center;">2</td> <td>32x32</td> </tr> <tr> <td style="text-align: center;">1</td> <td>16x16</td> </tr> <tr> <td style="text-align: center;">0</td> <td>8x8</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>In VDENC mode, This field should be programmed as 0.</td> </tr> </tbody> </table>	Format:	U2	Value	Name	3	64x64	2	32x32	1	16x16	0	8x8	Programming Notes	In VDENC mode, This field should be programmed as 0.					
Format:	U2																				
Value	Name																				
3	64x64																				
2	32x32																				
1	16x16																				
0	8x8																				
Programming Notes																					
In VDENC mode, This field should be programmed as 0.																					
3	31	<p>sps_range_extension_enable_flag</p> <p>This flag represents both sps_extension_present_flag and sps_range_extension_flag flags in SPS. If enabled, sps_extension_present_flag=1 and sps_range_extension_flag = 0 or 1.</p> <p>If sps_range_extension_flag = 0, default values are set for range extension parameters.</p> <p>0 no range extension 1 range extension</p>																			

HCP_PIC_STATE	
	Decoder only feature
30	<p>transform_skip_rotation_enabled_flag</p> <p>0 no rotation (default)1 apply rotation to the intra 4x4 residual data block coded in transform skip.</p> <p>Decoder only feature</p>
29	<p>transform_skip_context_enabled_flag</p> <p>0 no context select for parsing sig_coeff_flag with skipped transform (default)1 apply context select.</p> <p>Decoder only feature</p>
28	<p>implicit_rdpkm_enabled_flag</p> <p>0 the residual modification process is not used for intra blocks in the CVS (default)1 apply this modification process for intra blocks using a transform bypass.</p> <p>Decoder only feature</p>
27	<p>explicit_rdpkm_enabled_flag</p> <p>0 the residual modification process is not used for inter blocks in the CVS (default)1 apply this modification process for inter blocks using a transform bypass.</p> <p>Decoder only feature</p>
26	<p>intra_smoothing_disabled_flag</p> <p>0 filtering process of neighbouring samples is not disabled (default)1 filtering process of neighbouring samples is unconditionally disabled for intra prediction.</p> <p>Decoder only feature</p>
25	<p>persistent_rice_adaptation_enabled_flag</p> <p>0 no previous sub-block state is used in Rice parameter derivation (default)1 use mode dependent statistics accumulated from previous sub-blocks to derive Rice parameter.</p> <p>Decoder only feature</p>
24	<p>cabac_bypass_alignment_enabled_flag</p> <p>0 no CABAC alignment process is used prior to bypass decoding (default)1 CABAC alignment process is used prior to bypass decoding coeff_sign_flag[] and coeff_abs_level_remaining[].</p> <p>Decoder only feature</p>
23	<p>cross_component_prediction_enabled_flag</p> <p>0 disable cross component prediction (default)1 Enable cross component prediction.</p> <p>Only for 4:4:4 can it be program to non-zero.</p> <p>Decoder only feature</p>
22:20	<p>Log2MaxTransformSkipSize</p> <p>Equal to log2_max_transform_skip_block_size_minus2 + 2</p> <p>It must less than or equal to Log2MaxTrafoSize.</p> <p>Decoder only feature</p>
19	<p>High Precision Offsets Enable Flag</p> <p>Specifies that weighted prediction offset values are signaled using a bit-depth-</p>

HCP_PIC_STATE													
	<p>dependent precision. 0 - Disable 1 - Enable</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: center; background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Decoder only</td> </tr> </tbody> </table>	Programming Notes		Decoder only									
Programming Notes													
Decoder only													
18	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
17:14	Reserved												
13:8	Reserved												
7:4	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
3	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
2	<p>InsertTestFlag</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>U1</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">1h</td> <td></td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th colspan="2" style="text-align: center; background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2"> <p>CABAC 0 Word Insertion Test Enable (Encoder Only) This bit will modify CABAC K equation so that a positive K value can be generated easily. This is done for validation purpose only. In normal usage this bit should be set to 0.</p> <p>Regular equation for generating 'K' value when CABAC 0 Word Insertion Test Enable is set to 0.</p> $K = \{ [((96 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs * 3) + 1023) / 1024] - bytes_in_picture \} / 3$ <p>Modified equation when CABAC 0 Word Insertion Test Enable bit set to 1.</p> $K = \{ [((1536 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs * 3) + 1023) / 1024] - bytes_in_picture \} / 3$ <p>Encoder only feature.</p> <p>This bit should be set to 0 in VDENC mode. This bit should be set to 0 in regular PAK mode.</p> </td> </tr> </tbody> </table>	Format:	U1	Value	Name	0h	[Default]	1h		Programming Notes		<p>CABAC 0 Word Insertion Test Enable (Encoder Only) This bit will modify CABAC K equation so that a positive K value can be generated easily. This is done for validation purpose only. In normal usage this bit should be set to 0.</p> <p>Regular equation for generating 'K' value when CABAC 0 Word Insertion Test Enable is set to 0.</p> $K = \{ [((96 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs * 3) + 1023) / 1024] - bytes_in_picture \} / 3$ <p>Modified equation when CABAC 0 Word Insertion Test Enable bit set to 1.</p> $K = \{ [((1536 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs * 3) + 1023) / 1024] - bytes_in_picture \} / 3$ <p>Encoder only feature.</p> <p>This bit should be set to 0 in VDENC mode. This bit should be set to 0 in regular PAK mode.</p>	
Format:	U1												
Value	Name												
0h	[Default]												
1h													
Programming Notes													
<p>CABAC 0 Word Insertion Test Enable (Encoder Only) This bit will modify CABAC K equation so that a positive K value can be generated easily. This is done for validation purpose only. In normal usage this bit should be set to 0.</p> <p>Regular equation for generating 'K' value when CABAC 0 Word Insertion Test Enable is set to 0.</p> $K = \{ [((96 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs * 3) + 1023) / 1024] - bytes_in_picture \} / 3$ <p>Modified equation when CABAC 0 Word Insertion Test Enable bit set to 1.</p> $K = \{ [((1536 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs * 3) + 1023) / 1024] - bytes_in_picture \} / 3$ <p>Encoder only feature.</p> <p>This bit should be set to 0 in VDENC mode. This bit should be set to 0 in regular PAK mode.</p>													
1	<p>CurPicIsI</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>U1</td> </tr> </table> <p>Specifies that the current picture is comprised solely of I slices and that there are no P or B slices in the picture.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 30%; text-align: center;">Value</th> <th style="width: 70%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> </tr> </tbody> </table>	Format:	U1	Value	Name								
Format:	U1												
Value	Name												

HCP_PIC_STATE																					
	0	<table border="1"> <tr> <td style="width: 100px;"></td> <td>Current picture has at least one P or B slice</td> </tr> </table> <table border="1" style="background-color: #e6f2ff;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2"> This bit should be set to "0". Note: The value of "1" setting ("Current picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly. In VDENC mode, this bit should be set to 0. </td> </tr> </table>		Current picture has at least one P or B slice	Programming Notes		This bit should be set to "0". Note: The value of "1" setting ("Current picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly. In VDENC mode, this bit should be set to 0.														
	Current picture has at least one P or B slice																				
Programming Notes																					
This bit should be set to "0". Note: The value of "1" setting ("Current picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly. In VDENC mode, this bit should be set to 0.																					
	0	<p>ColPicIsl</p> <table border="1"> <tr> <td style="width: 100px;">Format:</td> <td>U1</td> </tr> </table> <p>Specifies that the collocated picture is comprised solely of I slices and that there are no P or B slices in the picture.</p> <table border="1" style="background-color: #e6f2ff;"> <tr> <th style="width: 100px;">Value</th> <th>Name</th> </tr> <tr> <td>0</td> <td>Collocated picture has at least one P or B slice</td> </tr> </table> <table border="1" style="background-color: #e6f2ff;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2"> This bit should be set to "0". Note: The value of "1" setting ("Collocated picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly. </td> </tr> </table>	Format:	U1	Value	Name	0	Collocated picture has at least one P or B slice	Programming Notes		This bit should be set to "0". Note: The value of "1" setting ("Collocated picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly.										
Format:	U1																				
Value	Name																				
0	Collocated picture has at least one P or B slice																				
Programming Notes																					
This bit should be set to "0". Note: The value of "1" setting ("Collocated picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly.																					
4	31:28	<p>Reserved</p> <table border="1"> <tr> <td style="width: 100px;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																	
Format:	MBZ																				
	27	<p>CU packet structure</p> <table border="1" style="background-color: #e6f2ff;"> <tr> <th style="width: 50px;">Value</th> <th style="width: 50px;">Name</th> <th>Description</th> </tr> <tr> <td>0</td> <td>VME</td> <td>CU packet structure is in output format of HW VME which is 1/2 CL per CU</td> </tr> <tr> <td>1</td> <td>ExtEnc</td> <td>CU packet structure is in output format of non HW VME (External Enc/SW) which is 1 CL per CU</td> </tr> </table> <table border="1" style="background-color: #e6f2ff;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> <tr> <td colspan="2">Value Must be zero only.</td> </tr> <tr> <td colspan="2">ExtEnc CU packet structure is not validated.</td> </tr> <tr> <td colspan="2">In VDENC mode, this bit should always be set to 0.</td> </tr> </table>	Value	Name	Description	0	VME	CU packet structure is in output format of HW VME which is 1/2 CL per CU	1	ExtEnc	CU packet structure is in output format of non HW VME (External Enc/SW) which is 1 CL per CU	Programming Notes		Encoder Only		Value Must be zero only.		ExtEnc CU packet structure is not validated.		In VDENC mode, this bit should always be set to 0.	
Value	Name	Description																			
0	VME	CU packet structure is in output format of HW VME which is 1/2 CL per CU																			
1	ExtEnc	CU packet structure is in output format of non HW VME (External Enc/SW) which is 1 CL per CU																			
Programming Notes																					
Encoder Only																					
Value Must be zero only.																					
ExtEnc CU packet structure is not validated.																					
In VDENC mode, this bit should always be set to 0.																					
	26	<p>strong_intra_smoothing_enable_flag</p> <table border="1"> <tr> <td style="width: 100px;">Format:</td> <td>U1</td> </tr> </table>	Format:	U1																	
Format:	U1																				
	25	<p>transquant_bypass_enable_flag</p> <table border="1"> <tr> <td style="width: 100px;">Format:</td> <td>Enable</td> </tr> </table> <table border="1" style="background-color: #e6f2ff;"> <tr> <th style="width: 100px;">Value</th> <th>Name</th> <th>Description</th> </tr> </table>	Format:	Enable	Value	Name	Description														
Format:	Enable																				
Value	Name	Description																			

HCP_PIC_STATE			
	0	Disable	cu_transquant_bypass is not supported
	1	Enable	cu_transquant_bypass is supported
24	Reserved		
	Format:		MBZ
23	amp_enabled_flag		
	Format:		Enable
	Value	Name	Description
	0	Disable	Asymmetric motion partitions cannot be used in coding tree blocks.
	1	Enable	Support asymmetric motion partitions, i.e. PartMode equal to PART_2NxN _U , PART_2NxN _D , PART_nLx2N, or PART_nRx2N.
	Programming Notes		
	In VDENC mode, this bit should be set to 1.		
22	transform_skip_enabled_flag		
	Format:		Enable
	Value	Name	Description
	0	Disable	transform_skip_flag is not supported in the residual coding
	1	Enable	transform_skip_flag is supported
21	BottomField		
	Format:		U1
	Value	Name	Description
	0	Bottom Field	
	1	Top Field	
	Programming Notes		
	Must be zero for encoder only		
20	FieldPic		
	Format:		U1
	Value	Name	Description
	0	Video Frame	
	1	Video Field	
	Programming Notes		
	Must be zero for encoder only.		

HCP_PIC_STATE			
	<p>This must be programmed to "0" when pps_curr_pic_ref_enabled_flag (SCC IBC mode) is "1".</p> <p>To handle SCC IBC field mode, each field (top/bottom) should be decoded as separate frame and the two fields should be interlaced as a separate step.</p>		
19	<p>weighted_pred_flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, this bit should be set to 1.</p>	Format:	U1
Format:	U1		
18	<p>weighted_bipred_flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, this bit should be set to 1.</p>	Format:	U1
Format:	U1		
17	<p>tiles_enabled_flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>This is an enable flag to enable tiling.</p> <p>Only Screen-Extended Main 4:4:4 and Screen-Extended Main 4:4:4 10 allows both <code>tiles_enabled_flag</code> and <code>entropy_coding_sync_enabled_flag</code> to be ON at the same time. All other current supported profile only allowed one of them to be ON.</p>	Format:	U1
Format:	U1		
16	<p>entropy_coding_sync_enabled_flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Not used in encoder mode</p> <p style="text-align: center;">Programming Notes</p> <p>Only Screen-Extended Main 4:4:4 and Screen-Extended Main 4:4:4 10 allows both <code>tiles_enabled_flag</code> and <code>entropy_coding_sync_enabled_flag</code> to be ON at the same time. All other current supported profile only allowed one of them to be ON.</p>	Format:	U1
Format:	U1		
15	<p>loop_filter_across_tiles_enabled_flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>For the encoder this bit must be set to zero (hardware restriction). Tile support is added, this field can be 0 or 1.</p> <p>Must be turned off if Partial Frame Update Mode is enabled</p>	Format:	U1
Format:	U1		
14	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
13	<p>sign_data_hiding_flag</p>		

HCP_PIC_STATE											
		Format: Enable									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>Specifies that sign bit hiding is disabled.</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>Specifies that sign bit hiding is enabled.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Specifies that sign bit hiding is disabled.	1	Enable	Specifies that sign bit hiding is enabled.
Value	Name	Description									
0	Disable	Specifies that sign bit hiding is disabled.									
1	Enable	Specifies that sign bit hiding is enabled.									
		<p style="text-align: center;">Programming Notes</p> <p>Currently not supported in encoder, so must be set to 0 for encoding session.</p>									
12:10		<p>log2_parallel_merge_level_minus2</p> <p>Format: U3</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Programming Notes</th> </tr> </thead> <tbody> <tr> <td>[0,4]</td> <td>Valid Range</td> <td>The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to Log2CtbSizeYCbLog2SizeY - 2, inclusive.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>For encoder, always set to 0 (Intel restriction).</p>	Value	Name	Programming Notes	[0,4]	Valid Range	The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to Log2CtbSizeYCbLog2SizeY - 2, inclusive.			
Value	Name	Programming Notes									
[0,4]	Valid Range	The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to Log2CtbSizeYCbLog2SizeY - 2, inclusive.									
9		<p>constrained_intra_pred_flag</p> <p>Format: U1</p>									
8		<p>pcm_loop_filter_disable_flag</p> <p>Format: U1</p> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, this bit is always set to 1.</p>									
7:6		<p>diff_cu_qp_delta_depth (or named as max_dqp_depth)</p> <p>Format: U2</p> <p style="text-align: center;">Programming Notes</p> <p>cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 or 3 is supported in PAK standalone and VDenc modes.</p> <p>cu_qp_delta_enabled_flag/max_dqp_depth: 1/0: has cu qp delta. (cu depth <= max_dqp_depth) will have cu qp delta coded. Only allow QP change across Slices, no change across LCU or CU.</p> <p>In VDenc mode, Max_DQP_Level can be 0 or 3.</p>									
5		<p>cu_qp_delta_enabled_flag</p> <p>Format: U1</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>Does not allow QP change at CU or LCU level, the same QP is used for</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Does not allow QP change at CU or LCU level, the same QP is used for			
Value	Name	Description									
0	Disable	Does not allow QP change at CU or LCU level, the same QP is used for									

HCP_PIC_STATE		
		the entire slice. Max_DQP_Level = 0 (i.e. diff_cu_qp_delta_depath = 0).
	1	Enable Allow QP change at CU level. MAX_DQP_level can be >0.
	Programming Notes	
	cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 or 3 is supported for PAK standalone and VDEnc modes.	
	In VDENC mode, this field should always be set to 1.	
	4	pcm_enabled_flag
	Format:	U1
	Programming Notes	
	In VDENC mode, this bit is always set to zero.	
	3	sample_adaptive_offset_enabled_flag
	Format:	U1
	Programming Notes	
	Restriction: HW does not support SAO filtering for LCU size 16x16 Should be set to 0 if slice_sao_luma_flag==0 and slice_sao_chroma_flag==0. It is illegal to program Slice_size_conformance ON when SAO is enabled in both first and second passes HW supports 8bit SAO and source format nv12 only. Must be off for Partial Frame Update Mode in encoder mode	
	Restriction: HW does not support SAO filtering for LCU size 16x16 Should be set to 0 if slice_sao_luma_flag==0 and slice_sao_chroma_flag==0. HW supports both 8 and 10b SAO in single pass	
	2:0	Reserved
	Format:	MBZ
5	31	Reserved
	30	Reserved
	Format:	MBZ
	29:27	bit_depth_luma_minus8
	Format:	U3
	This specifies the number of bit allow for Luma pixels. In 8 bit mode, this must be set to 0. Encoder: Supports bit depths 8, 10 and 12 only. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK i.e. the source pixel depth should be less than or equal to PAK bit depth.	
	Value	Name Description
	0	luma_8bit
	1	luma_9bit Only HEVC decoder supports 9 bits luma.

HCP_PIC_STATE		
		HEVC encoder does not supports 9 bits luma.
2	luma_10bit	
3	luma_11bit	<p>☐</p> <p>HEVC SCC does not support 11 bits Luma Also only HEVC decoder (non-SCC) support 11 bits Luma HEVC encoder (non-SCC) does not support 11 bits Luma</p>
4	luma_12bit	<p>☐</p> <p>HEVC SCC does not support 12bits Luma</p>
Programming Notes		
In VDENC mode, This field can only support a value of 0 or 2.		
26:24	bit_depth_chroma_minus8	
	Format:	U3
<p>This specifies the number of bit allow for Chroma pixels. In 8 bit mode, this must be set to 0. Encoder: Supports bit depths 8, 10 and 12 only. And also it must be same as Luma. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK. i.e. The source pixel depth should be less than or equal to the PAK bit depth.</p>		
	Value	Name
		Description
0	chroma_8bit	
1	chroma_9bit	Only HEVC decoder supports 9 bits chroma. HEVC encoder does not supports 9 bits chroma.
2	chroma_10bit	
3	chroma_11bit	<p>☐</p> <p>HEVC SCC does not support 11 bits chroma Also only HEVC decoder (non-SCC) support 11 bits chroma HEVC encoder (non-SCC) does not support 11 bits chroma</p>
4	chroma_12bit	<p>☐</p> <p>HEVC SCC does not support 12bits Luma</p>
Programming Notes		
In VDENC mode, this field should match bits 29:27 of this DW.		
23:20	pcm_sample_bit_depth_luma_minus1	
	Format:	U4
Description		
Encoder: Supports bit depths 8, 10 and 12 for PCM. Must be same as pixel depth. 12bit not validated		
Encoder supports 8, 10 and 12 bits. Must be same for luma and Chroma Cb/Cr. Decoder will support up to 12 bits.		
19:16	pcm_sample_bit_depth_chroma_minus1	

HCP_PIC_STATE																
		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U4</td> </tr> <tr> <td colspan="2" style="text-align: center;">Description</td> </tr> <tr> <td colspan="2">Bit depth must be same as luma for encoder.</td> </tr> <tr> <td colspan="2">Encoder support 8, 10 and 12 bits. Must be the same for luma and chroma Cb and Cr Decode will support up to 12 bits.</td> </tr> </table>	Format:	U4	Description		Bit depth must be same as luma for encoder.		Encoder support 8, 10 and 12 bits. Must be the same for luma and chroma Cb and Cr Decode will support up to 12 bits.							
Format:	U4															
Description																
Bit depth must be same as luma for encoder.																
Encoder support 8, 10 and 12 bits. Must be the same for luma and chroma Cb and Cr Decode will support up to 12 bits.																
	15:13	max_transform_hierarchy_depth_inter(or named as tu_max_depth_inter) <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U3</td> </tr> <tr> <td colspan="2">Maximum TU split depths for inter blocks</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">For encoder, always set to 2 to allow max 2 levels of split. For more splitting rely on CU split to match the content (Intel restriction)</td> </tr> </table>	Format:	U3	Maximum TU split depths for inter blocks		Programming Notes		For encoder, always set to 2 to allow max 2 levels of split. For more splitting rely on CU split to match the content (Intel restriction)							
Format:	U3															
Maximum TU split depths for inter blocks																
Programming Notes																
For encoder, always set to 2 to allow max 2 levels of split. For more splitting rely on CU split to match the content (Intel restriction)																
	12:10	max_transform_hierarchy_depth_intra (or named as tu_max_depth_intra) <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U3</td> </tr> <tr> <td colspan="2">Maximum TU split depth for intra blocks.</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">For encoder, always set to 2 to allow max 2 levels of split. For more splitting, rely on CU split to match the content (Intel restriction).</td> </tr> </table>	Format:	U3	Maximum TU split depth for intra blocks.		Programming Notes		For encoder, always set to 2 to allow max 2 levels of split. For more splitting, rely on CU split to match the content (Intel restriction).							
Format:	U3															
Maximum TU split depth for intra blocks.																
Programming Notes																
For encoder, always set to 2 to allow max 2 levels of split. For more splitting, rely on CU split to match the content (Intel restriction).																
	9:5	pic_cr_qp_offset <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S4</td> </tr> <tr> <td colspan="2">Valid range -12 to +12</td> </tr> </table>	Format:	S4	Valid range -12 to +12											
Format:	S4															
Valid range -12 to +12																
	4:0	pic_cb_qp_offset <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S4</td> </tr> <tr> <td colspan="2">Valid range -12 to +12</td> </tr> </table>	Format:	S4	Valid range -12 to +12											
Format:	S4															
Valid range -12 to +12																
6	31:30	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ												
	Format:	MBZ														
29	Load Slice Pointer Flag <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Enable</td> </tr> <tr> <td colspan="2">LoadBitStreamPointerPerSlice (Encoder-only)</td> </tr> <tr> <td colspan="2">To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> <td style="text-align: center;">Description</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Disable</td> <td>Load BitStream Pointer only once for the first slice of a frame.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Enable</td> <td>Load/reload BitStream Pointer only once for the each slice, reload the</td> </tr> </table>	Format:	Enable	LoadBitStreamPointerPerSlice (Encoder-only)		To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.		Value	Name	Description	0	Disable	Load BitStream Pointer only once for the first slice of a frame.	1	Enable	Load/reload BitStream Pointer only once for the each slice, reload the
Format:	Enable															
LoadBitStreamPointerPerSlice (Encoder-only)																
To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.																
Value	Name	Description														
0	Disable	Load BitStream Pointer only once for the first slice of a frame.														
1	Enable	Load/reload BitStream Pointer only once for the each slice, reload the														

HCP_PIC_STATE		
		start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field.
	Programming Notes	
	Must be zero for encoder	
28:27	Reserved	
	Format:	MBZ
26	FrameSzUnderStatusEn - FrameBitRateMinReportMask	
	Format:	Enable
	This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin.	
	Value	Name Description
	0	Disable Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register.
	1	Enable Set bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit Rate Minimum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit in HCP_IMAGE_STATUS_MASK register
	Programming Notes	
	Encoder Only	
	In VDENC mode, set this bit to zero always.	
25	FrameSzOverStatusEn - FrameBitRateMaxReportMask	
	Format:	Enable
	This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax.	
	Value	Name Description
	0	Disable Do not update bit 1 of HCP_IMAGE_STATUS control register.
	1	Enable Set bit 1 of HCP_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit Rate Maximum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit in HCP_IMAGE_STATUS_MASK register
	Programming Notes	
	Encoder Only	
	In VDENC mode, set this bit to zero always.	

HCP_PIC_STATE										
24	LCUMaxBitStatusEn - LCUMaxSizeReportMask Format: Enable This is a mask bit controlling if the condition of any LCU in the frame exceeds LCUMaxSize.									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>Do not update bit 0 of HCP_IMAGE_STATUS control register.</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>Set bit 0 of HCP_IMAGE_STATUS control register if the total bit counter for the current LCU is greater than the LCU Conformance Max size limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Do not update bit 0 of HCP_IMAGE_STATUS control register.	1	Enable	Set bit 0 of HCP_IMAGE_STATUS control register if the total bit counter for the current LCU is greater than the LCU Conformance Max size limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register.
	Value	Name	Description							
	0	Disable	Do not update bit 0 of HCP_IMAGE_STATUS control register.							
1	Enable	Set bit 0 of HCP_IMAGE_STATUS control register if the total bit counter for the current LCU is greater than the LCU Conformance Max size limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register.								
Programming Notes										
Encoder Only										
23:19	Reserved Format: MBZ									
18:17	Reserved Format: MBZ									
16	NonFirstPassFlag Format: Enable This signals the current pass is not the first pass. It will imply designate HW behavior.									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>If it is initial-Pass, this bit is set to 0.</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>For subsequent passes, this bit is set to 1.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	If it is initial-Pass, this bit is set to 0.	1	Enable	For subsequent passes, this bit is set to 1.
	Value	Name	Description							
	0	Disable	If it is initial-Pass, this bit is set to 0.							
1	Enable	For subsequent passes, this bit is set to 1.								
Programming Notes										
Encoder Only										
15:0	LCU Max BitSize Allowed Format: U16 This field specifies the Max LCU Bit Size allowed in according to the spec conformance. However, driver can program a different value from the spec for other encoding purpose. If LCU Limit exceeds, LCUMaxBitStatus in MMIO would be set to 1 If LCU Limit exceeds, LCUMaxBitStatus in MMIO would be set to 1 and also in CU Statistics Streamout, LCU Limit exceed field would be set to 1 at the last CU of a LCU									
	Programming Notes									
	Encoder Only									
	For now, only the lower 18 bits are sufficient to support 8 to 12-bit 444. The extra									

HCP_PIC_STATE											
		upper bits are reserved for 16-bit pixel depth in future,									
7 Programming Notes: Encoder Only	31	FrameBitrateMaxUnit									
		Format: U1									
		This field is the Frame Bitrate Maximum Limit Units.									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Byte</td> <td>32byte unit</td> </tr> <tr> <td>1</td> <td>Kilo Byte</td> <td>4kbyte unit</td> </tr> </tbody> </table>	Value	Name	Description	0	Byte	32byte unit	1	Kilo Byte	4kbyte unit
		Value	Name	Description							
	0	Byte	32byte unit								
	1	Kilo Byte	4kbyte unit								
	Programming Notes										
	Encoder Only										
	30:14	Reserved	Format: MBZ								
13:0		FrameBitRateMax									
		Format: U14									
		This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value.									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[0-16383]</td> <td></td> <td>When FrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.</td> </tr> <tr> <td>[0-16383]</td> <td></td> <td>When FrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.</td> </tr> </tbody> </table>	Value	Name	Description	[0-16383]		When FrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.	[0-16383]		When FrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
Value	Name	Description									
[0-16383]		When FrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.									
[0-16383]		When FrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.									
8	31	FrameBitrateMinUnit									
		Format: U1									
		This field is the Frame Bitrate Minimum Limit Units.									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Byte</td> <td>32byte unit</td> </tr> <tr> <td>1</td> <td>Kilo Byte</td> <td>4kbyte unit</td> </tr> </tbody> </table>	Value	Name	Description	0	Byte	32byte unit	1	Kilo Byte	4kbyte unit
		Value	Name	Description							
	0	Byte	32byte unit								
	1	Kilo Byte	4kbyte unit								
	Programming Notes										
	Encoder Only										
	30:14	Reserved	Format: MBZ								
13:0		FrameBitRateMin									
		Format: U14									
		This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit									

HCP_PIC_STATE																		
		<p>determines minimum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count happen to be below this value.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[0-16383]</td> <td></td> <td>When FrameBitrateMinUnit=0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.</td> </tr> <tr> <td>[0-16383]</td> <td></td> <td>When FrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Encoder Only</p>	Value	Name	Description	[0-16383]		When FrameBitrateMinUnit=0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.	[0-16383]		When FrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.							
Value	Name	Description																
[0-16383]		When FrameBitrateMinUnit=0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.																
[0-16383]		When FrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.																
9	31	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																	
	30:16	<p>FrameBitRateMaxDelta</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Always</td> </tr> <tr> <td>Format:</td> <td>U15</td> </tr> </table> <p>This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>[Default]</td> <td></td> </tr> <tr> <td>[0-32767]</td> <td></td> <td>When FrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.</td> </tr> <tr> <td>[0-32767]</td> <td></td> <td>When FrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Encoder Only</p>	Exists If:	//Always	Format:	U15	Value	Name	Description	0	[Default]		[0-32767]		When FrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.	[0-32767]		When FrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.
Exists If:	//Always																	
Format:	U15																	
Value	Name	Description																
0	[Default]																	
[0-32767]		When FrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.																
[0-32767]		When FrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.																
	15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																	
	14:0	<p>FrameBitRateMinDelta</p> <table border="1"> <tr> <td>Format:</td> <td>U15</td> </tr> </table> <p>This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> </tbody> </table>	Format:	U15	Value	Name	Description											
Format:	U15																	
Value	Name	Description																

HCP_PIC_STATE																			
	<table border="1"> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">[Default]</td> <td></td> </tr> <tr> <td style="text-align: center;">[0-32767]</td> <td></td> <td>When FrameBitrateMinUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.</td> </tr> <tr> <td style="text-align: center;">[0-32767]</td> <td></td> <td>When FrameBitrateMinUnit = 1, this field is measured in unit of 4K bytes (1K= 1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.</td> </tr> <tr> <td colspan="3" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="3">HW requires the following condition: FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.</td> </tr> <tr> <td colspan="3">Encoder Only</td> </tr> </table>	0	[Default]		[0-32767]		When FrameBitrateMinUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.	[0-32767]		When FrameBitrateMinUnit = 1, this field is measured in unit of 4K bytes (1K= 1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.	Programming Notes			HW requires the following condition: FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.			Encoder Only		
0	[Default]																		
[0-32767]		When FrameBitrateMinUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.																	
[0-32767]		When FrameBitrateMinUnit = 1, this field is measured in unit of 4K bytes (1K= 1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.																	
Programming Notes																			
HW requires the following condition: FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.																			
Encoder Only																			
10..11	63:0	<table border="1"> <tr> <td colspan="2">FrameDeltaQpMax</td> </tr> <tr> <td>Format:</td> <td style="text-align: center;">FrameDeltaQp</td> </tr> <tr> <td colspan="2">Range: [0:MAX_QP_DELTA]</td> </tr> <tr> <td colspan="2">Frame level delta QP which should be used in case FrameSize - FrameBitRateMax in the range of $((\text{DeltaQpMaxRange}[n] * \text{FrameBitRateMaxDelta} \gg 5))$, $\text{DeltaQpMaxRange}[n+1] * \text{FrameBitRateMaxDelta} \gg 5$.</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">If n == 7, DeltaQpMaxRange is infinity. Format: 8 bit sign-magnitude, 8bit-> Range: [0: 51] 10bit-> Range: [0:63] 12bit-> Range: [0:75]</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	FrameDeltaQpMax		Format:	FrameDeltaQp	Range: [0:MAX_QP_DELTA]		Frame level delta QP which should be used in case FrameSize - FrameBitRateMax in the range of $((\text{DeltaQpMaxRange}[n] * \text{FrameBitRateMaxDelta} \gg 5))$, $\text{DeltaQpMaxRange}[n+1] * \text{FrameBitRateMaxDelta} \gg 5$.		Programming Notes		If n == 7, DeltaQpMaxRange is infinity. Format: 8 bit sign-magnitude, 8bit-> Range: [0: 51] 10bit-> Range: [0:63] 12bit-> Range: [0:75]		Encoder Only				
FrameDeltaQpMax																			
Format:	FrameDeltaQp																		
Range: [0:MAX_QP_DELTA]																			
Frame level delta QP which should be used in case FrameSize - FrameBitRateMax in the range of $((\text{DeltaQpMaxRange}[n] * \text{FrameBitRateMaxDelta} \gg 5))$, $\text{DeltaQpMaxRange}[n+1] * \text{FrameBitRateMaxDelta} \gg 5$.																			
Programming Notes																			
If n == 7, DeltaQpMaxRange is infinity. Format: 8 bit sign-magnitude, 8bit-> Range: [0: 51] 10bit-> Range: [0:63] 12bit-> Range: [0:75]																			
Encoder Only																			
12..13	63:0	<table border="1"> <tr> <td colspan="2">FrameDeltaQpMin</td> </tr> <tr> <td>Format:</td> <td style="text-align: center;">FrameDeltaQp</td> </tr> <tr> <td colspan="2">Range: [0:MIN_QP_DELTA]</td> </tr> <tr> <td colspan="2">Frame level delta QP which should be used in case FrameSize - FrameBitRateMin in the range of $((\text{DeltaQpMinRange}[n] * \text{FrameBitRateMinDelta} \gg 5))$, $\text{DeltaQpMinRange}[n+1] * \text{FrameBitRateMinDelta} \gg 5$.</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">If n == 7, DeltaQpMinRange is infinity. Format: 8 bit sign-magnitude 8bit-> Range: [-51: 0]</td> </tr> </table>	FrameDeltaQpMin		Format:	FrameDeltaQp	Range: [0:MIN_QP_DELTA]		Frame level delta QP which should be used in case FrameSize - FrameBitRateMin in the range of $((\text{DeltaQpMinRange}[n] * \text{FrameBitRateMinDelta} \gg 5))$, $\text{DeltaQpMinRange}[n+1] * \text{FrameBitRateMinDelta} \gg 5$.		Programming Notes		If n == 7, DeltaQpMinRange is infinity. Format: 8 bit sign-magnitude 8bit-> Range: [-51: 0]						
FrameDeltaQpMin																			
Format:	FrameDeltaQp																		
Range: [0:MIN_QP_DELTA]																			
Frame level delta QP which should be used in case FrameSize - FrameBitRateMin in the range of $((\text{DeltaQpMinRange}[n] * \text{FrameBitRateMinDelta} \gg 5))$, $\text{DeltaQpMinRange}[n+1] * \text{FrameBitRateMinDelta} \gg 5$.																			
Programming Notes																			
If n == 7, DeltaQpMinRange is infinity. Format: 8 bit sign-magnitude 8bit-> Range: [-51: 0]																			

HCP_PIC_STATE																	
		10bit-> Range: [-63:0] 12bit-> Range: [-75:0] Encoder Only															
14..15	63:0	FrameDeltaQpMaxRange Format: FrameDeltaQpRange Range: [0:U8_MAX] Condition: FrameDeltaQpMaxRange[n] >= FrameDeltaQpMaxRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. <div style="text-align: center; background-color: #e6f2ff; padding: 5px;">Programming Notes</div> If n == 0, FrameDeltaQpMaxRange is zero. Encoder Only															
16..17	63:0	FrameDeltaQpMinRange Format: FrameDeltaQpRange Range: [0:U8_MAX] Condition: FrameDeltaQpMinRange[n] >= FrameDeltaQpMinRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. <div style="text-align: center; background-color: #e6f2ff; padding: 5px;">Programming Notes</div> If n == 0, FrameDeltaQpMinRange is zero. Encoder Only															
18	31:30	MinFrameSizeUnits Format: U2 This field is the Minimum Frame Size Units <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>4Kb</td> <td>Minimum Frame Size is in 4Kbytes.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>16Kb</td> <td>Minimum Frame Size is in 16Kbytes.</td> </tr> <tr> <td style="text-align: center;">2</td> <td>Compatibility Mode</td> <td>Minimum Frame Size is in 4bytes</td> </tr> <tr> <td style="text-align: center;">3</td> <td>16 Bytes</td> <td>Minimum Frame Size is 16 bytes.</td> </tr> </tbody> </table> <div style="text-align: center; background-color: #e6f2ff; padding: 5px;">Programming Notes</div> Encoder Only	Value	Name	Description	0	4Kb	Minimum Frame Size is in 4Kbytes.	1	16Kb	Minimum Frame Size is in 16Kbytes.	2	Compatibility Mode	Minimum Frame Size is in 4bytes	3	16 Bytes	Minimum Frame Size is 16 bytes.
Value	Name	Description															
0	4Kb	Minimum Frame Size is in 4Kbytes.															
1	16Kb	Minimum Frame Size is in 16Kbytes.															
2	Compatibility Mode	Minimum Frame Size is in 4bytes															
3	16 Bytes	Minimum Frame Size is 16 bytes.															
	29:16	Reserved Format: MBZ															

HCP_PIC_STATE										
	15:0	<p>MinFrameSize</p> <table border="1"> <tr> <td>Default Value:</td> <td>0</td> </tr> <tr> <td>Format:</td> <td>U16</td> </tr> </table> <p>Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. It is needed for CBR. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax. This field is reserved in Decode mode.</p> <table border="1"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> <tr> <td>Programmable range is $0..(2^{16}-1) * 2^{12}$ when MinFrameSizeUnits is 0. (4KB unit)</td> </tr> <tr> <td>Programmable range is $0..(2^{16}-1) * 2^{14}$ when MinFrameSizeUnits is 1. (16KB unit)</td> </tr> <tr> <td>Encoder Only</td> </tr> </table>	Default Value:	0	Format:	U16	Programming Notes	Programmable range is $0..(2^{16}-1) * 2^{12}$ when MinFrameSizeUnits is 0. (4KB unit)	Programmable range is $0..(2^{16}-1) * 2^{14}$ when MinFrameSizeUnits is 1. (16KB unit)	Encoder Only
	Default Value:	0								
	Format:	U16								
	Programming Notes									
	Programmable range is $0..(2^{16}-1) * 2^{12}$ when MinFrameSizeUnits is 0. (4KB unit)									
Programmable range is $0..(2^{16}-1) * 2^{14}$ when MinFrameSizeUnits is 1. (16KB unit)										
Encoder Only										
19	31	<p>Temporal MV pred disable</p> <p>Indicates Temporal MV pred/Surface is disabled This value should be set to 1 in Partial Frame Update Mode, to disable the Temporal MV prediction.</p> <table border="1"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> <tr> <td>This bit must be set to zero in encoder non-Partial Frame Update and decoder mode. SW must disable temporal MV in slice header for Partial Frame update Mode</td> </tr> </table>	Programming Notes	This bit must be set to zero in encoder non-Partial Frame Update and decoder mode. SW must disable temporal MV in slice header for Partial Frame update Mode						
	Programming Notes									
	This bit must be set to zero in encoder non-Partial Frame Update and decoder mode. SW must disable temporal MV in slice header for Partial Frame update Mode									
	30	<p>Partial Frame Update Mode</p> <table border="1"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> <tr> <td> Loop filter across slices and tiles should be turned off SAO filter should be turned off for entire frame Last row of a Tile can't be skip unless the Tile has all skip rows Last row of a Frame can't be skip unless the Frame has all skip rows and Tiling disabled for that frame (This frame level restriction does not apply when Tiling enabled) Temporal MV prediction should be off No Multipass allowed Note: A Tile row can have skip and nonSkip tiles Any Tile row can be Skip in a frame VDenc mode only </td> </tr> </table>	Programming Notes	Loop filter across slices and tiles should be turned off SAO filter should be turned off for entire frame Last row of a Tile can't be skip unless the Tile has all skip rows Last row of a Frame can't be skip unless the Frame has all skip rows and Tiling disabled for that frame (This frame level restriction does not apply when Tiling enabled) Temporal MV prediction should be off No Multipass allowed Note: A Tile row can have skip and nonSkip tiles Any Tile row can be Skip in a frame VDenc mode only						
Programming Notes										
Loop filter across slices and tiles should be turned off SAO filter should be turned off for entire frame Last row of a Tile can't be skip unless the Tile has all skip rows Last row of a Frame can't be skip unless the Frame has all skip rows and Tiling disabled for that frame (This frame level restriction does not apply when Tiling enabled) Temporal MV prediction should be off No Multipass allowed Note: A Tile row can have skip and nonSkip tiles Any Tile row can be Skip in a frame VDenc mode only										
29:28	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
27:26	<p>NumberOfLCUsInNormal Slice size conformance Mode</p> <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <p>Indicates Max number of LCUs can be issued in Normal Mode of SliceSizeConformance. This is a performance feature to keep HW pipeline full.</p>	Format:	U2							
Format:	U2									

HCP_PIC_STATE									
	<table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center; background-color: #e6f2ff;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Programming Notes		Encoder Only					
Programming Notes									
Encoder Only									
25	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
24	<p>SSE Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field indicates if SSE statistics generation is enabled in the PAK. If enabled, the classID in the CU packet is used along with the SSE thresholds in the picture state to generate the SSE frame statistics. When enabled, the PAK will also write and read source pixels (similar to the reference pixels in the LCU ILDB streamout) to generate the SSE distortion.</p> <p>This parameter is valid to both HEVC and VP9. In HEVC, it is valid for LCU64 and LCU32 modes.</p> <p>The global streamout flag (PAK Pipeline streamout enable) must be enabled if this bit is set to 1</p> <p>Encoder only</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center; background-color: #e6f2ff;">Programming Notes</th> </tr> <tr> <td colspan="2">Valid only in VDenc mode. In VDENC mode this bit should be set to one.</td> </tr> <tr> <td colspan="2">In Partial Frame Update mode, This feature can't be enabled in VDENC mode.</td> </tr> </table>	Format:	Enable	Programming Notes		Valid only in VDenc mode. In VDENC mode this bit should be set to one.		In Partial Frame Update mode, This feature can't be enabled in VDENC mode.	
Format:	Enable								
Programming Notes									
Valid only in VDenc mode. In VDENC mode this bit should be set to one.									
In Partial Frame Update mode, This feature can't be enabled in VDENC mode.									
23:18	<p>slice_pic_parameter_set_id</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U6</td> </tr> </table> <p>This parameter is used in Dynamic slice mode when inserting slice headers. slice_pic_parameter_set_id specifies the value of pps_pic_parameter_set for the PPS in use. The value of slice_pic_parameter_set_id shall be in the range of 0 to 63, inclusive.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center; background-color: #e6f2ff;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U6	Programming Notes		Encoder Only			
Format:	U6								
Programming Notes									
Encoder Only									
17	<p>NalUnitTypeFlag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U1</td> </tr> </table> <p>This parameter is used in Dynamic slice mode when inserting slice headers. (derived as (nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center; background-color: #e6f2ff;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U1	Programming Notes		Encoder Only			
Format:	U1								
Programming Notes									
Encoder Only									
16	<p>first_slice_segment_in_pic_flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U1</td> </tr> </table> <p>This parameter is used in Dynamic slice mode when inserting slice headers. first_slice_segment_in_pic_flag equal to 1 specifies that the slice segment is the first slice segment of the picture in decoding order. first_slice_segment_in_pic_flag equal to 0 specifies that the slice segment is not the first slice segment of the picture in decoding order.</p> <p>Note: This bit must be always 1 so HW can update in subsequent slices.</p>	Format:	U1						
Format:	U1								

HCP_PIC_STATE																	
	<table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Programming Notes		Encoder Only													
Programming Notes																	
Encoder Only																	
15	<p>no_output_of_prior_pics_flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>This parameter is used in Dynamic slice mode when inserting slice headers. no_output_of_prior_pics_flag affects the output of previously-decoded pictures in the decoded picture buffer after the decoding of an IDR or a BLA picture that is not the first picture in the bitstream</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U1	Programming Notes		Encoder Only											
Format:	U1																
Programming Notes																	
Encoder Only																	
14	<p>PAK Dynamic Slice Mode Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Enable</td> </tr> </table> <p>This field controls whether PAK will check will check the Slice Size Threshold in Bytes parameter for bits overflow and terminate the slice. This mode is called Dynamic Slice Mode.</p> <p>When this field is disabled, PAK is in Static Slice Mode. It uses the LastCtbOfSlice Flag in the HCP_PAK_OBJECT to terminate the slice.</p> <p>Note: HW infer "dependent_slice_segments_enabled_flag" to be 0 when PAK Dynamic Slice Mode Enable = 1</p> <table border="1" style="width: 100%; margin-left: 20px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>Disable [Static Slice Mode]</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>Enable [Dynamic Slice Mode]</td> </tr> </tbody> </table> <table border="1" style="width: 100%; margin-top: 10px;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">The max Tile Size allowed is 5k when Partial Frame Update Mode and Dynamic Slice Mode are enabled</td> </tr> <tr> <td colspan="2">This bit can be set to '1' only for HEVC codec.</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	Enable	Value	Name	0h	Disable [Static Slice Mode]	1h	Enable [Dynamic Slice Mode]	Programming Notes		The max Tile Size allowed is 5k when Partial Frame Update Mode and Dynamic Slice Mode are enabled		This bit can be set to '1' only for HEVC codec.		Encoder Only	
Format:	Enable																
Value	Name																
0h	Disable [Static Slice Mode]																
1h	Enable [Dynamic Slice Mode]																
Programming Notes																	
The max Tile Size allowed is 5k when Partial Frame Update Mode and Dynamic Slice Mode are enabled																	
This bit can be set to '1' only for HEVC codec.																	
Encoder Only																	
13:8	<p>RhoDomainFrameLevelQP</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>This QP is used for RhoDomain Frame level statistics</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U6	Programming Notes		Encoder Only											
Format:	U6																
Programming Notes																	
Encoder Only																	
7	<p>Fractional QP adjustment enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Enable</td> </tr> </table> <table border="1" style="width: 100%; margin-top: 10px;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Enables Fractional QP adjustment feature in PAK.</td> </tr> <tr> <td colspan="2">Valid for LCU sizes 64x64 and 32x32 only.</td> </tr> <tr> <td colspan="2">Must be disabled for LCU 16x16</td> </tr> </table>	Format:	Enable	Programming Notes		Enables Fractional QP adjustment feature in PAK.		Valid for LCU sizes 64x64 and 32x32 only.		Must be disabled for LCU 16x16							
Format:	Enable																
Programming Notes																	
Enables Fractional QP adjustment feature in PAK.																	
Valid for LCU sizes 64x64 and 32x32 only.																	
Must be disabled for LCU 16x16																	

HCP_PIC_STATE			
	<p>If enabled then cu_qp_delta_enable_flag must to be set to 1 Encoder Only</p> <p>In VDENC mode, set this bit to 1.</p>		
	<p>6 RhoDomain Rate Control Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field indicates if RhoDomain statistics generation is enabled in the PAK. The RhoDomain AverageMacroblockQP is used by the PAK along with the CU QP to compute the statistics. The global streamout flag (PAK Pipeline Streamout Enable) needs to be enabled when RhoDomain Rate Control is enabled in-order to stream out the rho-domain statistics..</p> <p>RhoDomain statistics will have non_zero coeffs when Force_zero_coeff feature enabled</p> <p style="text-align: center;">Programming Notes</p> <p>Encoder Only</p> <p>In VDENC mode, set this bit to 1.</p>	Format:	Enable
Format:	Enable		
	<p>5:3 Fractional QP Offset</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U3</td> </tr> </table> <p>Start segment from the top of the Frame where increased Quantization parameter is added. Within two 64x64 LCUs/four 32x32 LCU rows, the total number LCUs are equally split into 8 segments (i.e.each segment has (FrameWidthInLCU « (4 LCUSize) + 7)»3 LCUs).</p> <p style="text-align: center;">Programming Notes</p> <p>Encoder Only</p>	Format:	U3
Format:	U3		
	<p>2:0 Fractional QP Input</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U3</td> </tr> </table> <p>In a set of 8 segments, Qp is incremented by 1 for F_qp segments.</p> <p style="text-align: center;">Programming Notes</p> <p>Encoder Only</p>	Format:	U3
Format:	U3		
20	<p>31:7 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
	<p>6 IntraTuCountBasedRDOQdisable</p> <p style="text-align: center;">Description</p> <p>When this field is set (disabled), RDOQ will be disabled based on whether intraTU counting (accumulated at a 4x4 level for all TU sizes -- 4x4/8x8/16x16/32x32) is greater than the threshold. Frame level RDOQ enable flag must be set if this is set. When it is unset (i.e. 0), RDOQ enable/disable is decided based on frame level RDOQ</p>		

HCP_PIC_STATE		
		enable flag.
	5:0	Reserved Format: MBZ
21	31:0	Slice Size Threshold in Bytes Format: U32 When a slice exceeds this value in bytes, hardware will end current slice and insert a new slice boundary. <div style="text-align: center;">Programming Notes</div> Slice Size Threshold in Bytes cannot be less than 600 Bytes when Partial Frame Update mode is ON Encoder Only
22	31:0	Target Slice Size in Bytes Format: U32 This field indicates the target slice size in Bytes for slice size conformance feature. When the actual slice size exceeds "Target slice size in Bytes", HW sets "Slice Overflow Occurred" bit in the PAK Frame Statistics. <div style="text-align: center;">Programming Notes</div> Encoder Only
23	31:16	Class0_SSE_Threshold1 Format: U16 This field specifies the upper bound threshold for Class0 Zone1 to classify the per-4x4sblk SSE statistics. Class0_SSE_Threshold_0 < per-4x4sblk SSE <= Class0_SSE_Threshold_1 fall under Class0 Zone1. Class0_SSE_Threshold_1 < per-4x4sblk SSE fall under Class0 Zone2. <div style="text-align: center;">Programming Notes</div> Encoder Only
	15:0	Class0_SSE_Threshold0 Format: U16 This field specifies the upper bound threshold for Class0 Zone0 to classify the per-4x4sblk SSE statistics. per-4x4sblk SSE <= Class0_SSE_Threshold_0 fall under Class0 Zone0. <div style="text-align: center;">Programming Notes</div> Encoder Only
24..31	255:0	SSE thresholds for Class1-8 Format: U256 SSE thresholds for Class 1-8, see DW 21 (SSE Class 0 thresholds) for format. <div style="text-align: center;">Programming Notes</div> Encoder Only

HCP_PIC_STATE		
32	31:30	Reserved Format: MBZ
	29:25	cb_qp_offset_list[5] Format: S4 Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Programming Notes</div> Decoder Only
	24:20	cb_qp_offset_list[4] Format: S4 Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Programming Notes</div> Decoder Only
	19:15	cb_qp_offset_list[3] Format: S4 Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Programming Notes</div> Decoder Only
	14:10	cb_qp_offset_list[2] Format: S4 Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Programming Notes</div> Decoder Only
	9:5	cb_qp_offset_list[1] Format: S4 Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Programming Notes</div> Decoder Only
	4:0	cb_qp_offset_list[0]

HCP_PIC_STATE											
	<table border="1"> <tr> <td>Format:</td> <td>S4</td> </tr> <tr> <td colspan="2"> Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) </td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Decoder Only</td> </tr> </table>	Format:	S4	Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)		Programming Notes		Decoder Only			
Format:	S4										
Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)											
Programming Notes											
Decoder Only											
33	<table border="1"> <tr> <td>31:30</td> <td>Reserved</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	31:30	Reserved	Format:	MBZ						
	31:30	Reserved									
	Format:	MBZ									
	<table border="1"> <tr> <td>29:25</td> <td>cr_qp_offset_list[5]</td> </tr> <tr> <td>Format:</td> <td>S4</td> </tr> <tr> <td colspan="2"> Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) </td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Decoder Only</td> </tr> </table>	29:25	cr_qp_offset_list[5]	Format:	S4	Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)		Programming Notes		Decoder Only	
	29:25	cr_qp_offset_list[5]									
Format:	S4										
Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)											
Programming Notes											
Decoder Only											
<table border="1"> <tr> <td>24:20</td> <td>cr_qp_offset_list[4]</td> </tr> <tr> <td>Format:</td> <td>S4</td> </tr> <tr> <td colspan="2"> Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) </td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Decoder Only</td> </tr> </table>	24:20	cr_qp_offset_list[4]	Format:	S4	Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)		Programming Notes		Decoder Only		
24:20	cr_qp_offset_list[4]										
Format:	S4										
Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)											
Programming Notes											
Decoder Only											
<table border="1"> <tr> <td>19:15</td> <td>cr_qp_offset_list[3]</td> </tr> <tr> <td>Format:</td> <td>S4</td> </tr> <tr> <td colspan="2"> Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) </td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Decoder Only</td> </tr> </table>	19:15	cr_qp_offset_list[3]	Format:	S4	Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)		Programming Notes		Decoder Only		
19:15	cr_qp_offset_list[3]										
Format:	S4										
Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)											
Programming Notes											
Decoder Only											
<table border="1"> <tr> <td>14:10</td> <td>cr_qp_offset_list[2]</td> </tr> <tr> <td>Format:</td> <td>S4</td> </tr> <tr> <td colspan="2"> Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) </td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Decoder Only</td> </tr> </table>	14:10	cr_qp_offset_list[2]	Format:	S4	Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)		Programming Notes		Decoder Only		
14:10	cr_qp_offset_list[2]										
Format:	S4										
Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)											
Programming Notes											
Decoder Only											
<table border="1"> <tr> <td>9:5</td> <td>cr_qp_offset_list[1]</td> </tr> <tr> <td>Format:</td> <td>S4</td> </tr> </table>	9:5	cr_qp_offset_list[1]	Format:	S4							
9:5	cr_qp_offset_list[1]										
Format:	S4										

HCP_PIC_STATE																	
		<p>Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="2">Programming Notes</td> </tr> <tr> <td colspan="2">Decoder Only</td> </tr> </table>	Programming Notes		Decoder Only												
Programming Notes																	
Decoder Only																	
	4:0	<p>cr_qp_offset_list[0]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S4</td> </tr> </table> <p>Offsets used in the derivation of Qp'_{cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="2">Programming Notes</td> </tr> <tr> <td colspan="2">Decoder Only</td> </tr> </table>	Format:	S4	Programming Notes		Decoder Only										
Format:	S4																
Programming Notes																	
Decoder Only																	
34	31	<p>intra_boundary_filtering_disabled_flag This specifies that the intra boundary filtering process is unconditionally disabled for intra prediction. Decoder only (Encoder default to "0")</p>															
	30:29	<p>motion_vector_resolution_control_idc This controls the presence and inference of the use_integer_mv_flag that specifies the resolution of motion vectors for inter prediction. Decoder only (Encoder default to "00")</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No integer MV for the frame</td> <td></td> </tr> <tr> <td>1</td> <td>Only integer MV for the frame</td> <td></td> </tr> <tr> <td>2</td> <td>Adaptive integer MV for the frame</td> <td>Slice signal use_inter_mv_flag will indicate if the slice will use integer MV or not</td> </tr> <tr> <td>3</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0	No integer MV for the frame		1	Only integer MV for the frame		2	Adaptive integer MV for the frame	Slice signal use_inter_mv_flag will indicate if the slice will use integer MV or not	3	Reserved	
	Value	Name	Description														
	0	No integer MV for the frame															
1	Only integer MV for the frame																
2	Adaptive integer MV for the frame	Slice signal use_inter_mv_flag will indicate if the slice will use integer MV or not															
3	Reserved																
28	<p>pps_curr_pic_ref_enabled_flag This bit specifies that a picture referring to the PPS may be included in a reference picture list of a slice of the picture itself. This applies to both decoder and encoder</p>																
27	<p>residual_adaptive_colour_transform_enabled_flag This bit specifies that an adaptive color transform may be applied to the residual in the decoding process. When ChromaArrayType is not equal to 3, residual_adaptive_colour_transform_enabled_flag shall be equal to 0. Decoder only (Encoder default to "0")</p>																

HCP_PIC_STATE	
26	<p>pps_slice_act_qp_offsets_present_flag This bit specifies that slice_act_y_qp_offset, slice_act_cb_qp_offset, slice_act_cr_qp_offset are present in the slice header. Decoder only (Encoder default to "0")</p>
25:20	<p>pps_act_y_offset_plus5 This is used to determine the offsets that are applied to the quantization parameter values qP derived in clause 8.6.2 for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. The variable PpsActQpOffsetY is set equal to pps_act_y_qp_offset_plus5. It is a requirement of bitstream conformance that the values of PpsActQpOffsetY shall be in the range of 12 to +12, inclusive. [Valid Range is from -7 to 17] Decoder only (Encoder default to "0")</p>
19:14	<p>pps_act_cb_qp_offset_plus5 This is used to determine the offsets that are applied to the quantization parameter values qP derived in clause 8.6.2 for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. The variable PpsActQpOffsetCb is set equal to pps_act_cb_qp_offset_plus5. It is a requirement of bitstream conformance that the values of PpsActQpOffsetCb shall be in the range of 12 to +12, inclusive. [Valid Range is from -7 to 17] Decoder only (Encoder default to "0")</p>
13:8	<p>pps_act_cr_qp_offset_plus3 This is used to determine the offsets that are applied to the quantization parameter values qP derived in clause 8.6.2 for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. The variable PpsActQpOffsetCr is set equal to pps_act_cr_qp_offset_plus3. It is a requirement of bitstream conformance that the values of PpsActQpOffsetCr shall be in the range of 12 to +12, inclusive. [Valid Range is from -9 to 15] Decoder only (Encoder default to "0")</p>
7	<p>pps_deblocking_filter_disabled_flag pps_deblocking_filter_disabled_flag equal to 1 specifies that the operation of deblocking filter is not applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present. pps_deblocking_filter_disabled_flag equal to 0 specifies that the operation of the deblocking filter is applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present.</p>
6	<p>deblocking_filter_override_enabled_flag deblocking_filter_override_enabled_flag equal to 1 specifies the presence of deblocking_filter_override_flag in the slice headers for pictures referring to the PPS.</p>

HCP_PIC_STATE																							
		deblocking_filter_override_enabled_flag equal to 0 specifies the absence of deblocking_filter_override_flag in the slice headers for pictures referring to the PPS.																					
	5:3	Reserved																					
		Format:	MBZ																				
	2:0	IBC Motion Compensation Buffer Reference IDC This indicates which one of the eight Motion Compensation buffer is mapped to current picture for IBC. Motion Compensation will use this picture to write and read IBC data This applies to both decoder and encoder																					
35	31	palette_mode_enabled_flag This specifies that the palette mode may be used for intra blocks. This applies to both decoder and encoder																					
	30	monochrome_palette_flag <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; background-color: #e1eef6;">Description</th> </tr> </thead> <tbody> <tr> <td>This specifies that the pictures that refer to this PPS are monochrome. This must be set to 0. Decoder only [Encoder default to 0]</td> </tr> </tbody> </table>		Description	This specifies that the pictures that refer to this PPS are monochrome. This must be set to 0. Decoder only [Encoder default to 0]																		
	Description																						
This specifies that the pictures that refer to this PPS are monochrome. This must be set to 0. Decoder only [Encoder default to 0]																							
29:28	IBC configuration IBC configuration is used configure Intra block copy. - Disable Intra block copy. - Limit Intra block copy from Left blocks only. - Allow full range of Intra block copy as specified in spec.																						
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; background-color: #e1eef6;">Value</th> <th style="text-align: center; background-color: #e1eef6;">Name</th> <th style="text-align: center; background-color: #e1eef6;">Description</th> <th style="text-align: center; background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td></td> <td>When IBC configuration in VDENC mode is set to 3, Intra block search includes top and left regions. In decoder mode, When SCC is enabled this field should be set to 3.</td> <td></td> </tr> <tr> <td style="text-align: center;">2</td> <td></td> <td></td> <td>This is invalid value.</td> </tr> <tr> <td style="text-align: center;">1</td> <td></td> <td>When IBC configuration in fixed function encoder (VDENC) mode is set to 1, Intra block search includes only left region.</td> <td></td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">[Default]</td> <td>When IBC configuration is 0, intra block copy is disabled and it applies for both Fixed function encoder and decoder.</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	Programming Notes	3		When IBC configuration in VDENC mode is set to 3, Intra block search includes top and left regions. In decoder mode, When SCC is enabled this field should be set to 3.		2			This is invalid value.	1		When IBC configuration in fixed function encoder (VDENC) mode is set to 1, Intra block search includes only left region.		0	[Default]	When IBC configuration is 0, intra block copy is disabled and it applies for both Fixed function encoder and decoder.		
Value	Name	Description	Programming Notes																				
3		When IBC configuration in VDENC mode is set to 3, Intra block search includes top and left regions. In decoder mode, When SCC is enabled this field should be set to 3.																					
2			This is invalid value.																				
1		When IBC configuration in fixed function encoder (VDENC) mode is set to 1, Intra block search includes only left region.																					
0	[Default]	When IBC configuration is 0, intra block copy is disabled and it applies for both Fixed function encoder and decoder.																					
	27:24	luma_bit_depth_entry_minus8 This specifies the bit depth of the luma component of the entries of the palette																					

HCP_PIC_STATE				
		<p>predictor initializer BitDepthEntryY as follows: $\text{BitDepthEntryY} = 8 + \text{luma_bit_depth_entry_minus8}$ It is a requirement of bitstream conformance that the value of BitDepthEntryY shall be equal to the BitDepthY of the SPS referred to by this PPS. Valid Range is from 0 to 4. This applies to both decoder and encoder</p>		
	23:20	<p>chroma_bit_depth_entry_minus8 This specifies the bit depth of the chroma components of the entries of the palette predictor initializer BitDepthEntryC as follows: $\text{BitDepthEntryC} = 8 + \text{chroma_bit_depth_entry_minus8}$ It is a requirement of bitstream conformance that the value of BitDepthEntryC shall be equal to BitDepthC of the SPS referred to by this PPS. Valid Range is from 0 to 4. This applies to both decoder and encoder</p>		
	19	<p>IBC Motion Vector Error Handling Disable Illegal IBC motion vector (pointing to future) can cause hang to HW. Error handling is needed to avoid hang. This bit disable the error handling if needed</p>		
	18:17	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
	16:10	<p>delta_palette_max_predictor_size This specifies the difference between the maximum allowed palette predictor size and the maximum allowed palette size. It is a requirement of bitstream conformance that the value of delta_palette_max_predictor_size shall be equal to 0 when palette_max_size is equal to 0. Valid value is from 0 to 64 $\text{PaletteMaxPredictorSize} = \text{palette_max_size} + \text{delta_palette_max_predictor_size}$ This applies to both decoder and encoder PaletteMaxPredictorSize will be up to 96 for encoder</p>		
	9:7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
	6:0	<p>palette_max_size This specifies the maximum allowed palette size. Valid value is from 0 to 64 This applies to both decoder and encoder. Always 64 for encoder</p>		
36	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			

HCP_PIC_STATE				
37	31:16	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	RDOQIntraTUThreshold <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; background-color: #e1eef6;">Description</th> </tr> </thead> <tbody> <tr> <td> This parameter programs the threshold for the Intra Counting. This value is programmed as a multiple of 16. When checking against the IntraTU Count, this value needs to be $\ll 4$ (multiplied by 16) and then compared against the IntraTU count to trigger the condition for Intra RDOQ disable. The IntraTU Count is accumulated in units of 4x4. </td> </tr> </tbody> </table>	Description	This parameter programs the threshold for the Intra Counting. This value is programmed as a multiple of 16. When checking against the IntraTU Count, this value needs to be $\ll 4$ (multiplied by 16) and then compared against the IntraTU count to trigger the condition for Intra RDOQ disable. The IntraTU Count is accumulated in units of 4x4.	
Description				
This parameter programs the threshold for the Intra Counting. This value is programmed as a multiple of 16. When checking against the IntraTU Count, this value needs to be $\ll 4$ (multiplied by 16) and then compared against the IntraTU count to trigger the condition for Intra RDOQ disable. The IntraTU Count is accumulated in units of 4x4.				
38	31:16	RDOQIntra32x32TUThreshold		
	15:0	RDOQIntra16x16TUThreshold		
39..40	63:0	SSEThresholds for Class9 ..10 SSE thresholds for Class 9-10, see DW 21 (SSE Class 0 thresholds) for format		

HCP_PIPE_BUF_ADDR_STATE

HCP_PIPE_BUF_ADDR_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>This state command provides the memory base addresses for the row store buffer and reconstructed picture output buffers required by the HCP.</p> <p>This is a picture level state command and is shared by both encoding and decoding processes.</p>			
Programming Notes			
All pixel surface addresses must be 4K byte aligned. There is a max of 8 Reference Picture BufferAddresses, and all share the same third address DW in specifying 48-bit address.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
	26:23	Media Instruction Opcode	
Default Value:		7h Codec/Engine Name	
Format:		OpCode	
Codec/Engine Name = HCP = 7h			
22:16	Media Instruction Command		
	Default Value:	2h HCP_PIPE_BUF_ADDR_STATE	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	66h		
	72h		
1..2	63:0	Decoded Picture	

HCP_PIPE_BUF_ADDR_STATE				
		<table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>Frame buffer address for the final decoded picture YUV output.</p>	Format:	SplitBaseAddress4KByteAligned
Format:	SplitBaseAddress4KByteAligned			
3	31:0	Decoded Picture Memory Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
4..5	63:0	Deblocking Filter Line Buffer <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the filter line buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
6	31:0	Deblocking Filter Line Buffer Memory Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
7..8	63:0	Deblocking Filter Tile Line Buffer <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile line buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
9	31:0	Deblocking Filter Tile Line Buffer Memory Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
10..11	63:0	Deblocking Filter Tile Column Buffer <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address of the tile column buffer (read/write) used by the Deblocking Filter.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
12	31:0	Deblocking Filter Tile Column Buffer Memory Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
13..14	63:0	Metadata Line Buffer <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address for the Metadata Line buffer.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
15	31:0	Metadata Line Buffer Memory Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
16..17	63:0	Metadata Tile Line Buffer <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address for the Metadata Tile Line buffer.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
18	31:0	Metadata Tile Line Buffer Memory Address Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
19..20	63:0	Metadata Tile Column Buffer <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Base address for the Metadata Tile Column buffer.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			

HCP_PIPE_BUF_ADDR_STATE		
21	31:0	Metadata Tile Column Buffer Memory Address Attributes Format: MemoryAddressAttributes
22..23	63:0	SAO Line Buffer Format: SplitBaseAddress64ByteAligned Base address for the SAO Line buffer.
24	31:0	SAO Line Buffer Memory Address Attributes Format: MemoryAddressAttributes
25..26	63:0	SAO Tile Line Buffer Format: SplitBaseAddress64ByteAligned Base address for the SAO Tile Line buffer.
27	31:0	SAO Tile Line Buffer Memory Address Attributes Format: MemoryAddressAttributes
28..29	63:0	SAO Tile Column Buffer Format: SplitBaseAddress64ByteAligned Base address for the SAO Tile Column buffer.
30	31:0	SAO Tile Column Buffer Memory Address Attributes Format: MemoryAddressAttributes
31..32	63:0	Current Motion Vector Temporal Buffer Format: SplitBaseAddress64ByteAligned Base address for the Current Motion Vector Temporal buffer.
33	31:0	Current Motion Vector Temporal Buffer Memory Address Attributes Format: MemoryAddressAttributes
34..35	63:0	Reserved Format: MBZ
36	31:0	Reserved Format: MBZ
37..52	511:0	Reference Picture Base Address (RefAddr[0-7]) Format: SplitBaseAddress64ByteAligned[8] Base address of the reference picture buffer. <div style="border: 1px solid black; background-color: #e6f2ff; padding: 5px; margin-top: 5px;"> <p style="text-align: center; margin: 0;">Programming Notes</p> <p style="margin: 0;">This Note is applicable in VDENC mode in IBC enabled case. NumRefIdxL0_minus1 in VDENC defines number of references used for encoding the frame. When IBC is enabled, the base address corresponding to current non-deblocked reconstructed picture should programmed to Reference Picture Base Address[NumRefIdxL0_minus1]</p> </div>

HCP_PIPE_BUF_ADDR_STATE		
53	31:0	Reference Picture Base Address Memory Address Attributes
		Format: MemoryAddressAttributes
		<p style="text-align: center;">Programming Notes</p> <p>Reference Picture Memory Compression Enables and Types (Media/Render) are moved to HCP_SURFACE_STATE. Separate 8 Memory Compression Enables and Types are added in HCP_SURFACE_STATE so each reference picture surfaces have its own separate bits. The memory compression enable and type bit in this DW are not used.</p>
54..55	63:0	Original Uncompressed Picture Source
		Format: SplitBaseAddress64ByteAligned
		<p>Buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding.</p> <p>This value is only valid in encoding mode.</p>
56	31:0	Original Uncompressed Picture Source Memory Address Attributes
		Format: MemoryAddressAttributes
57..58	63:0	Streamout Data Destination
		Exists If: //Decoder Only
		Format: SplitBaseAddress64ByteAligned
		Description
		Buffer address for outputting the per-block indirect data to memory when StreamOutEnable is set in the HCP_PIPE_MODE_SELECT command.
		Decoder does not use this buffer. For Encoder: this field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit. For Encoder: This surface is used to streamout CU records All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-block data.
59	31:0	Streamout Data Destination Memory Address Attributes
		Exists If: //Decoder Only
		Format: MemoryAddressAttributes
60..61	63:0	Decoded Picture Status/Error Buffer Base Address or Encoded slice size streamout Base Address
		Format: SplitBaseAddress64ByteAligned
<p>Decoder Mode : Specifies the 64 byte aligned buffer address for writing a single status/error cache-line sized record into memory when the Pic Status/Error Report Enable is set in the HCP_PIPE_MODE_SELECT command. The pic status/error record is written by hardware after the picture is decoded.</p> <p>Encoder Mode: This specifies 64 byte aligned buffer address for writing Slice size, when slice</p>		

HCP_PIPE_BUF_ADDR_STATE		
		size conformance is enabled.
62	31:0	Decoded Picture Status/Error Buffer Base Address Memory Address Attributes Format: MemoryAddressAttributes
63..64	63:0	LCU ILDB Streamout Buffer Format: SplitBaseAddress64ByteAligned Buffer address for writing ILDB parameter per LCU to memory when Deblocker Streamout Enable is set in the HCP_PIPE_MODE_SELECT Command. The ILDB MB control parameters are written by HW at the end of each reconstructed LCU. Only edge information is being streamed out.
65	31:0	LCU ILDB Streamout Buffer Memory Address Attributes Format: MemoryAddressAttributes
66..81	511:0	Collocated Motion Vector Temporal Buffer[0-7] Format: SplitBaseAddress64ByteAligned[8] Base address for the Collocated Motion Vector Temporal buffer.
82	31:0	Collocated Motion Vector Temporal Buffer[0-7] Memory Address Attributes Format: MemoryAddressAttributes
83..84	63:0	VP9 Probability Buffer Read/Write Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for VP9 Probability Buffer. Hardware reads in the probability for decode and write out the modified probability for future frames. Driver needs to program the Initial VP9 Probability for decoding the current frame. For Key Frame, it should contain the default Key Frame Probability. For non-Key Frame, it could be a default (non-Key) or one of the 8 Reference Buffers Probability. Driver must provide a valid Initial VP9 Probability buffer.
85	31:0	VP9 Probability Buffer Read/Write Memory Address Attributes Format: MemoryAddressAttributes
86..87	63:0	VP9 Segment ID Buffer Read/Write Specifies the 64 byte aligned buffer address for VP9 SegmentID buffer. This should contain the writeout SegmentID from previous frame and will be used to predict SegmentID for the current frame. Hardware will write out SegmentID of the current frame in the same address for the next frame.
88	31:0	VP9 Segment ID buffer Read/Write Memory Address Attributes Format: MemoryAddressAttributes
89..90	63:0	VP9 HVD Line Rowstore Buffer Read/Write Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for HVD Tile Rowstore Buffer (bitstream decoder).

HCP_PIPE_BUF_ADDR_STATE		
91	31:0	VP9 HVD Line Rowstore buffer Read/Write Memory Address Attributes Format: MemoryAddressAttributes
92..93	63:0	VP9 HVD Tile Rowstore Buffer Read/Write Format: SplitBaseAddress64ByteAligned
94	31:0	VP9 HVD Tile Rowstore buffer Read/Write Memory Address Attributes Format: MemoryAddressAttributes
95..96	63:0	SAO Rowstore Buffer Base Address Specifies the 64 byte aligned buffer address for Rowstoring of SAO parameters in encoder mode
97	31:0	SAO Rowstore Buffer Read/Write Memory Address Attributes Format: MemoryAddressAttributes
98..99	63:0	Frame Statistics Streamout Data Destination Buffer Base Address Format: SplitBaseAddress64ByteAligned <div style="border: 1px solid black; padding: 5px; text-align: center; background-color: #e6f2ff;"> Description </div> Specifies the 64 byte aligned buffer address for outputting the frame statistics data to memory. The statistics are mainly SliceSize conformance, SSE, RhoDomain and CU parameters. Decoder does not use this buffer.
100	31:0	Frame Statistics Streamout Data Destination buffer (attributes) Read/Write Format: MemoryAddressAttributes
101..102	63:0	SSE Source Pixel RowStore Buffer Base Address Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for storing the source pixels for SSE. SSE metrics in the PAK are computed using post loop-filtered pixels or post SAO, if SAO is enabled.
103	31:0	SSE Source Pixel RowStore buffer (attributes) Read/Write Format: MemoryAddressAttributes
104..105	63:0	HCP Scalability Slice State Buffer Base Address Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for storing slice state information on HEVC/VP9 Scalable mode for decode. This is needed since CABAC and BE pass will be separated and BE pass needs to have slice state information as well. This buffer is only used in HEVC Scalable Decode Mode Only (Virtual Tile on both CABAC and Recon Pass)
106	31:0	HCP Scalability Slice State Buffer (attributes) Read/Write Format: MemoryAddressAttributes
107..108	63:0	HCP Scalability CABAC Decoded Syntax Elements Buffer Base Address

HCP_PIPE_BUF_ADDR_STATE				
		<table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned buffer address for storing CABAC Decoded Syntax Element on HEVC/VP9 Scalable mode for decode</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
109	31:0	HCP Scalability CABAC Decoded Syntax Elements Buffer (attributes) Read/Write <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
110..111	63:0	Motion Vector Upper Right Column Store Buffer Base Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned buffer address for storing upper right Motion Vector on HEVC /VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode. This buffer is only used on HEVC Scalable Decode Only (Virtual Tile on both CABAC and Recon pass)</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
112	31:0	Motion Vector Upper Right Column Store Buffer (attributes) Read/Write <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
113..114	63:0	Intra Prediction Upper Right Column Store Buffer Base Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned buffer address for storing upper right Intra Prediction Pixel on HEVC /VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
115	31:0	Intra Prediction Upper Right Column Store Buffer (attributes) Read/Write <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
116..117	63:0	Intra Prediction Left Recon Column Store Buffer Base Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned buffer address for storing left column Intra Prediction Pixel on HEVC /VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode.</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			
118	31:0	Intra Prediction Left Recon Column Store Buffer (attributes) Read/Write <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
119..120	63:0	HCP Scalability CABAC Decoded Syntax Elements Buffer Max Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned maximum address for the HCP Scalability CABAC Decoded Syntax Elements Buffer. This address shall either be 0 or larger than HCP Scalability CABAC Decoded Syntax Elements Buffer Base Address. If this address is 0, the upper bound is considered disable and HW will NOT check for upper bound. Hardware shall only write to memory address less than this address (unless address is 0 which is disabled). Hardware will not write to memory address larger than or equal to address (unless address is 0 which is disabled)</p>	Format:	SplitBaseAddress64ByteAligned
Format:	SplitBaseAddress64ByteAligned			

HCP_PIPE_MODE_SELECT

HCP_PIPE_MODE_SELECT			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The workload for the HCP is based upon a single frame decode. There are no states saved between frame decodes in the HCP. Once the bit stream DMA is configured with the HCP_BSD_OBJECT command, and the bitstream is presented to the HCP, the frame decode will begin.</p> <p>The HCP_PIPE_MODE_SELECT command is responsible for general pipeline level configuration that would normally be set once for a single stream encode or decode and would not be modified on a frame workload basis.</p> <p>This is a picture level state command and is shared by both encoding and decoding processes.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = HCP = 7h	
22:16	Media Instruction Command		
	Default Value:	0h HCP_PIPE_MODE_SELECT	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	5h	Value_5	
1	31:24	Reserved	
	23	Reserved	

HCP_PIPE_MODE_SELECT

22:20	Reserved					
	Format:	MBZ				
	19	Reserved				
		Format:	MBZ			
	18	Prefetch Disable				
<p>When memory compression is enabled, we are seeing drop in performance. To compensate loss of performance due to latencies, we are adding pre-fetch.</p> <p>This bit would disable prefetches if they cause unintended behavior.</p>						
17	Tile Based Engine					
	<p>This bit indicates HW works as a Tile Based Engine(as opposed to Frame based)meaning HW will flush out bitstream and streamout data to memory at the end of each Tile.</p> <p>Tiling must be enabled to set this bit to 1.</p> <p>If this bit set to 1, the Tile row CAN be repeated if needed.</p> <p>Only current tile row is allowed to be repeated otherwise SW has to repeat all Tile Rows starting from the top tile row in a frame.</p>					
16:15	Pipe working Mode					
	This programs the working mode for HCP pipe.					
	Value	Name	Description			
	00b	Legacy decoder/encoder mode (Single pipe)	This is for single pipe mode non-scalable mode. It is used by both decoder and encoder.			
	01b	CABAC FE only decode mode (Single CABAC pipe)	This is for the single CABAC FE only in decoder mode. This will be only run CABAC and streamout syntax element.			
	10b	Decoder BE only or Encoder mode (Scalable Multi-pipe)	This is for multiple-pipe scalable mode. In decoder, it is only on BE reconstruction. In encoder, it is for PAK.			
11b	Decoder Scalable mode with CABAC in real tiles (Scalable Multi-pipe)	This is for multiple-pipe scalable mode decoder mode in real tiles. CABAC and reconstruction will run together. Each pipes will run in real tiles vertically.	The real-tile/virtual tile decoding are supported for following features:			
			<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;"></td> <td style="width: 35%;">Virtual tile decoding</td> <td style="width: 35%;">Real tile decoding</td> </tr> <tr> <td>Rext HEVC (including main, main10)</td> <td>yes</td> <td>yes</td> </tr> </table>		Virtual tile decoding	Real tile decoding
	Virtual tile decoding	Real tile decoding				
Rext HEVC (including main, main10)	yes	yes				

HCP_PIPE_MODE_SELECT

					SCC HEVC	no	yes
					VP9	yes	no
					AV1	no	yes
14:13	Multi-Engine Mode This indicates the current pipe is in single pipe mode or if in scalable mode is in left/right/middle pipe in multi-engine mode.						
	Value	Name	Description				
	00b	Single Engine Mode or CABAC FE only decode mode	This is for single engine mode (legacy) OR CABAC FE only decode mode [] During HEVC Decoder Scalability Real Tile Mode, for the last phase, it is possible to have single tile column left. In this case, it should be programmed with pipe as a single engine mode (using this value). For example, for 9 tile column running on 4 pipes. The first two phases will use all 4 pipes and finish 8 tile column. The remaining one column will be processed as last third phase as single tile column.				
	01b	Pipe is the left engine in a Multi-engine mode	Current pipe is the most left engine while running in scalable multi-engine mode				
	10b	Pipe is the right engine in a Multi-engine mode	Current pipe is the most right engine while running in scalable multi-engine mode				
	11b	Pipe is one of the middle engine in a Multi-engine mode	Current pipe is in one of the middle engine while running in scalable multi-engine mode				
12	PAK Frame Level StreamOut enable This bit is valid if global bit PAK Pipeline Streamout Enable is set to 1. This bit is defined to use legacy tests on HW and it's valid for both hevc/vp9. Frame level streamouts consists of 3 parts: LCU Streamout (Set PAK Frame level streamout enable and PAK Pipeline Streamout Enable) SSE Streamout (Set SSE enable and PAK Pipeline Streamout Enable) RhoDomain Streamout (Set RhoDomain enable and PAK Pipeline Streamout Enable) If set to 1, HW will output LCU streamouts which are not validated. By default it should be '0'						
11	Reserved		Format:		MBZ		
10	VDEnc_Mode		Format:		Enable		
	This field indicates if PAK is working in legacy MBEnc mode or the VDEnc mode.						

HCP_PIPE_MODE_SELECT

	Value	Name	Description
	0h	MBEnc mode	PAK is working in legacy mode
	1h	VDEnc mode	PAK is working in VDEnc mode
9	Advanced Rate Control Enable		
	Format:		Enable
	It is only defined for encode.		
	Value	Name	Description
	0	Disable	Use the legacy HW generated delta QP for multipass
	1	Enable	Use the HuC based rate control (HW continues to generate the legacy delta QP and write to MMIO, but do not add to the final QP in the next pass)
	VP9: Scalability mode use only Advanced Rate Control for BRC (no HW involvement) HEVC: Scalability mode use only HW assisted Advanced Rate Control for BRC. HW assistance- HW adds delta QP for every CU in multipass.		
	Programming Notes		
	In VDENC mode, This bit should be set to 1 always. Only advanced rate control is supported in VDENC mode.		
8	Reserved		
	Format:		MBZ
7:5	Codec Standard Select		
	Value	Name	
	0	HEVC	
	1	VP9	
4	Reserved		
	Format:		MBZ
3	Pic Status/Error Report Enable		
	Format:		Enable
	Value	Name	Description
	0	Disable	Disable status/error reporting
	1	Enable	Status/Error reporting is written out once per picture. The Pic Status/Error Report ID in DWord3 along with the status/error status bits are packed into one cache line and written to the Status/Error Buffer address in the HCP_PIPE_BUF_ADDR_STATE command. Must be zero for encoder mode.
2	PAK Pipeline Streamout Enable		
	Format:		Enable
	Pipeline Streamout Enable is only defined for encode. It is ignored for decode.		

HCP_PIPE_MODE_SELECT																			
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th colspan="2" style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td colspan="2">Disable pipeline states and parameters streamout</td> </tr> <tr> <td style="text-align: center;">1</td> <td colspan="2">Enable pipeline states and parameters streamout</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="3" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="3">In VDENC mode, this field should be set to 1 always.</td> </tr> </tbody> </table>	Value	Name		0	Disable pipeline states and parameters streamout		1	Enable pipeline states and parameters streamout		Programming Notes			In VDENC mode, this field should be set to 1 always.				
Value	Name																		
0	Disable pipeline states and parameters streamout																		
1	Enable pipeline states and parameters streamout																		
Programming Notes																			
In VDENC mode, this field should be set to 1 always.																			
	1	<p>Deblocker Streamout Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Deblocker Streamout Enable not currently supported for Encode or Decode</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> <td>Disable deblocker-only parameter streamout</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Enable</td> <td>Enable deblocker-only parameter streamout</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	Description	0	Disable	Disable deblocker-only parameter streamout	1	Enable	Enable deblocker-only parameter streamout						
Format:	Enable																		
Value	Name	Description																	
0	Disable	Disable deblocker-only parameter streamout																	
1	Enable	Enable deblocker-only parameter streamout																	
	0	<p>Codec Select</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Decode</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Encode</td> </tr> </tbody> </table>	Format:	U1	Value	Name	0	Decode	1	Encode									
Format:	U1																		
Value	Name																		
0	Decode																		
1	Encode																		
2	31:0	<p>Media Soft-Reset Counter (per 1000 clocks)</p> <table border="1"> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>In decoder modes, this counter value specifies the number of clocks (per 1000) of GAC inactivity before a media soft-reset is applied to the HCP and HuC. If counter value is set to 0, the mediasoft-reset feature is disabled and no reset will occur.</p> <p>In encoder modes, this counter must be set to 0 to disable media soft reset. This feature is not supported for the encoder.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> </tr> </tbody> </table>	Format:	U32	Value	Name	0	Disable											
Format:	U32																		
Value	Name																		
0	Disable																		
3	31:0	<p>Pic Status/Error Report ID</p> <table border="1"> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>The Pic Status/Error Report ID is a unique 32-bit unsigned integer assigned to each picture status/error output. Must be zero for encoder mode.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>32-bit unsigned</td> <td>Unique ID Number</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Reserved</td> <td></td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="3" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="3">Software must program different Status/Error Buffer addresses between pictures; otherwise the hardware might overwrite previously written data.</td> </tr> </tbody> </table>	Format:	U32	Value	Name	Description	0	32-bit unsigned	Unique ID Number	1	Reserved		Programming Notes			Software must program different Status/Error Buffer addresses between pictures; otherwise the hardware might overwrite previously written data.		
Format:	U32																		
Value	Name	Description																	
0	32-bit unsigned	Unique ID Number																	
1	Reserved																		
Programming Notes																			
Software must program different Status/Error Buffer addresses between pictures; otherwise the hardware might overwrite previously written data.																			
4	31:0	Reserved																	

HCP_PIPE_MODE_SELECT										
		Format: MBZ								
5	31:0	Reserved Format: MBZ								
6	31:8	Reserved Format: MBZ								
	7:4	Reserved Format: MBZ								
	3	Frame reconstruction disable This bit disables writing out final reconstructed pixels to memory Normally used for B-frame as it's not used for reference purpose in encoder mode Default value should be '0'								
	2	HEVC Separate Tile Programming This indicates each tile should be programmed separately in single pipe mode. (Tile can have multiple slices. But in this case, the slice must end at end of tile so it does not affect this bit). If there are multiple tiles in a slice, the slice needs to split into each individual tiles and programmed each tiles separately). This should be set when the following is met: (tiles_enabled_flag == "1") && ((pps_curr_pic_ref_enabled_flag == "1") (palette_mode_enabled_flag == "1") (entropy_coding_sync_enabled_flag == 1))								
	1:0	Phase Indicator This is used to indicate whether this is first, middle or last phase of programming during Real-Tile Decoder Mode. Since HEVC can have up to 20 tile columns, maximum 10 phases are possible during 2 VDbbox scalable mode. This is used by hardware to know if the current programming is first or last phases. This field is ignored (programmed to 0) for other modes other than HEVC Real-Tile Decoder Mode.								
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>First Phase</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Middle Phase</td> </tr> <tr> <td style="text-align: center;">2</td> <td>Last Phase</td> </tr> </tbody> </table>	Value	Name	0	First Phase	1	Middle Phase	2	Last Phase
Value	Name									
0	First Phase									
1	Middle Phase									
2	Last Phase									

HCP_QM_STATE

HCP_QM_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The HCP_QM_STATE command loads the custom HEVC quantization tables into local RAM and may be issued up to 20 times: 3x Colour Component plus 2x intra/inter plus 4x SizeID minus 4 for the 32x32 chroma components.</p> <p>When the scaling_list_enable_flag is set to disable, the scaling matrix is still sent to the decoder, and with all entries programmed to the same value = 16.</p> <p>This is a picture level state command and is issued in both encoding and decoding processes.</p> <p>Dwords 2-17 form a table for the DCT coefficients, 4 8-bit coefficients/DWord.</p> <ul style="list-style-type: none"> • Size 4x4 for SizeID0, DWords 2-5. • Size 8x8 for SizeID1/2/3, DWords 2-17. 			
SizeID 0 (Table 4-10)			
	4x4	[31:24]	[23:16]
	[15:8]	[7:0]	
DWord 2	AC(0,3)	AC(0,2)	AC(0,1)
DWord 3	AC(1,3)	AC(1,2)	AC(1,1)
DWord 4	AC(2,3)	AC(2,2)	AC(2,1)
DWord 5	AC(3,3)	AC(3,2)	AC(3,1)
DWord 6	DC	AC(1,0)	AC(2,0)
DWord 7	AC(3,0)	AC(3,0)	AC(3,0)
SizeID 1, 2, 3 (Table 4-11)			
	8x8	[31:24]	[23:16]
	[15:8]	[7:0]	[31:24]
	[23:16]	[15:8]	[7:0]
DWord 3,2	AC(0,7)	AC(0,6)	AC(0,5)
DWord 5,4	AC(1,7)	AC(1,6)	AC(1,5)
DWord 7,6	AC(2,7)	AC(2,6)	AC(2,5)
DWord 9,8	AC(3,7)	AC(3,6)	AC(3,5)
DWord 11,10	AC(4,7)	AC(4,6)	AC(4,5)
DWord 13,12	AC(5,7)	AC(5,6)	AC(5,5)
DWord 15,14	AC(6,7)	AC(6,6)	AC(6,5)
DWord 17,16	AC(7,7)	AC(7,6)	AC(7,5)
DWord 18,17	AC(0,4)	AC(0,3)	AC(0,2)
DWord 19,18	AC(1,4)	AC(1,3)	AC(1,2)
DWord 20,19	AC(2,4)	AC(2,3)	AC(2,2)
DWord 21,20	AC(3,4)	AC(3,3)	AC(3,2)
DWord 22,21	AC(4,4)	AC(4,3)	AC(4,2)
DWord 23,22	AC(5,4)	AC(5,3)	AC(5,2)
DWord 24,23	AC(6,4)	AC(6,3)	AC(6,2)
DWord 25,24	AC(7,4)	AC(7,3)	AC(7,2)
DWord 26,25	AC(0,1)	AC(0,1)	AC(0,1)
DWord 27,26	AC(1,1)	AC(1,1)	AC(1,1)
DWord 28,27	AC(2,1)	AC(2,1)	AC(2,1)
DWord 29,28	AC(3,1)	AC(3,1)	AC(3,1)
DWord 30,29	AC(4,1)	AC(4,1)	AC(4,1)
DWord 31,30	AC(5,1)	AC(5,1)	AC(5,1)
DWord 32,31	AC(6,1)	AC(6,1)	AC(6,1)
DWord 33,32	AC(7,1)	AC(7,1)	AC(7,1)
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	

HCP_QM_STATE													
	<table border="1"> <tr> <td>Default Value:</td> <td>2h</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	2h	Format:	OpCode								
Default Value:	2h												
Format:	OpCode												
26:23	<p>Media Instruction Opcode</p> <table border="1"> <tr> <td>Default Value:</td> <td>7h Codec/Engine Name</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table> <p>Codec/Engine Name = HCP = 7h</p>	Default Value:	7h Codec/Engine Name	Format:	OpCode								
Default Value:	7h Codec/Engine Name												
Format:	OpCode												
22:16	<p>Media Instruction Command</p> <table border="1"> <tr> <td>Default Value:</td> <td>4h HCP_QM_STATE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	4h HCP_QM_STATE	Format:	OpCode								
Default Value:	4h HCP_QM_STATE												
Format:	OpCode												
15:12	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
11:0	<p>Dword Length</p> <table border="1"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>(Excludes Dwords 0, 1).</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>10h</td> <td></td> </tr> </tbody> </table>	Format:	=n	Value	Name	10h							
Format:	=n												
Value	Name												
10h													
1	<p>31:13</p>	Format:	MBZ										
Format:	MBZ												
	<p>12:5</p>	Format:	U8	Programming Notes		The DC Coefficient must be set to zero for SizeID 0 and 1.		The DC Coefficient must be set to scaling_list_dc_coef_minus8 + 8 for SizeID 2 and 3.					
Format:	U8												
Programming Notes													
The DC Coefficient must be set to zero for SizeID 0 and 1.													
The DC Coefficient must be set to scaling_list_dc_coef_minus8 + 8 for SizeID 2 and 3.													
	<p>4:3</p>	Format:	U2	Value	Name	0	Luma	1	Chroma Cb	2	Chroma Cr	3	Reserved
Format:	U2												
Value	Name												
0	Luma												
1	Chroma Cb												
2	Chroma Cr												
3	Reserved												
	<p>2:1</p>	Format:	U2										
Format:	U2												

HCP_QM_STATE																	
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>4x4</td> <td></td> </tr> <tr> <td>1</td> <td>8x8</td> <td></td> </tr> <tr> <td>2</td> <td>16x16</td> <td></td> </tr> <tr> <td>3</td> <td>32x32</td> <td>(Illegal Value for Color Component Chroma Cr and Cb.)</td> </tr> </tbody> </table>	Value	Name	Description	0	4x4		1	8x8		2	16x16		3	32x32	(Illegal Value for Color Component Chroma Cr and Cb.)
Value	Name	Description															
0	4x4																
1	8x8																
2	16x16																
3	32x32	(Illegal Value for Color Component Chroma Cr and Cb.)															
	0	<table border="1"> <thead> <tr> <th colspan="2">Prediction Type</th> </tr> </thead> <tbody> <tr> <td>Format:</td> <td>U1</td> </tr> <tr> <th>Value</th> <th>Name</th> </tr> <tr> <td>0</td> <td>Intra</td> </tr> <tr> <td>1</td> <td>Inter</td> </tr> </tbody> </table>	Prediction Type		Format:	U1	Value	Name	0	Intra	1	Inter					
Prediction Type																	
Format:	U1																
Value	Name																
0	Intra																
1	Inter																
2..17	511:0	QuantizerMatrix															

HCP_REF_IDX_STATE

HCP_REF_IDX_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>This is a slice level command used in both encoding and decoding processes. For decoder, it is issued with the HCP_BSD_OBJECT command.</p> <p>Unlike AVC, HEVC allows 16 reference idx entries in each of the L0 and L1 list for a progressive picture. Hence, a max total 32 reference idx in both lists together. The same when the picture is a field picture. Regardless the number of reference idx entries, there are only max 8 reference pictures exist at any one time. Multiple reference idx can point to the same reference picture and can optionally pic a top or bottom field or frame.</p> <p>For P-Slice, this command is issued only once, representing L0 list. For B-Slice, this command can be issued up to two times, one for L0 list and one for L1 list.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = HCP = 7h	
22:16	Media Instruction Command		
	Default Value:	12h HCP_REF_IDX_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	10h		
1	31:5	Reserved	

HCP_REF_IDX_STATE								
		Format: MBZ						
	4:1	num_ref_idx_l[RefPicListNum]_active_minus1 Format: U4 num_ref_idx_l[RefPicListNum]_active_minus1 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0-14]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0-14]			
Value	Name							
[0-14]								
	0	RefPicListNum Format: U1 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%; text-align: center;">Value</th> <th style="width: 70%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reference Picture List 0</td> </tr> <tr> <td>1</td> <td>Reference Picture List 1</td> </tr> </tbody> </table>	Value	Name	0	Reference Picture List 0	1	Reference Picture List 1
Value	Name							
0	Reference Picture List 0							
1	Reference Picture List 1							
2..17	511:0	Entries Format: HCP_REF_LIST_ENTRY[16]						

HCP_SFC_LOCK

HCP_SFC_LOCK				
Source:	BSpec			
Length Bias:	2			
Description				
<p>This command is used for VD/VE box to communicate with SFC before the start of any SFC workload. VD/VE uses this command to make sure that it has the ownership of SFC pipe before running workload with SFC since SFC is shared between VD/VE on a frame level.</p> <p>For VD(MFX)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC.</p> <p>For VD(HCP)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	Pipeline		
		Default Value:	2h Media	
		Format:	OpCode	
	26:23	Media Command Opcode		
		Format:	OpCode	
		Value	Name	Description
		9h	Media HCP+SFC Mode [Default]	For VD(HCP)+SFC mode, only decoder mode is allowed. Encoder mode cannot use SFC
22:21	SubOpcodeA			
	Default Value:	0h Common		
	Format:	OpCode		
20:16	SubOpcodeB			
	Default Value:	0h SFC Lock		
	Format:	OpCode		
15:12	Reserved			
	Format:	MBZ		
11:0	DWord Length			
	Default Value:	0h Excludes DWord (0,1)		
	Format:	=n		
	Total Length - 2			
1	31:0	Pre-Scaled Output Surface Output Enable		



HCP_SFC_LOCK

Format:	SFC_PRESCALED_OUTPUT_SURFACE_OUTPUT_ENABLE
---------	--

HCP_SFC_STATE

HCP_SFC_STATE			
Source:	BSpec		
Length Bias:	2		
This command is sent from VDBOX/HCP/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
	26:23	Media Command Opcode	
		Default Value:	9h Media HCP+SFC Mode
	22:21	SubOpcodeA	
Default Value:		0h Common	
20:16	SubOpcodeB		
	Default Value:	1h SFC_State	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	2Bh Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..49	1567:0	SFC State Body	
		Format:	SFC_STATE_BODY



HCP_SLICE_STATE

HCP_SLICE_STATE		
Source:	VideoCS	
Length Bias:	2	
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>This is a slice level command used in both encoding and decoding processes. For decoder, it is issued with the HCP_BSD_OBJECT command.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	Pipeline Type
		Default Value: 2h Format: OpCode
	26:23	Media Instruction Opcode
		Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	Media Instruction Command
Default Value: 14h HCP_SLICE_STATE Format: OpCode		
15:12	Reserved	
11:0	Dword Length	
	Format: =n (Excludes Dwords 0, 1).	
	Value	Name
	9h	
7h		
1	31:26	Reserved
	Format: MBZ	
25:16	SliceStartCtbY or (slice_start_lcu_y encoder)	
	Format: U10	
		Description

HCP_SLICE_STATE					
	<p>Specifies the starting row address of the first coding tree block in the current slice.</p> <p>The following applies to Real Tile Decoding Mode: (1) This should be programmed with the starting row address of the current slice segment. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles to be programmed separately. In this case, this should be programmed with the starting row address of the current tile.</p>				
15:10	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ				
9:0	<p>SliceStartCtbX or (slice_start_lcu_x encoder)</p> <table border="1"> <tr> <td>Format:</td> <td>U10</td> </tr> </table> <p style="text-align: center;">Description</p> <p>Specifies the starting column address of the first coding tree block in the current slice.</p> <p>The following applies to Real Tile Decoding Mode: (1) This should be programmed with the starting column address of the current slice segment. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be programmed with the starting column address of the current tile.</p> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, this bit should be programmed as zero</p>	Format:	U10		
Format:	U10				
2	<p>31:27 Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> <p>26:16 NextSliceStartCtbY or (next_slice_start_lcu_y encoder)</p> <table border="1"> <tr> <td>Format:</td> <td>U11</td> </tr> </table> <p>Specifies the starting row address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame height (at least +1).</p> <p style="text-align: center;">Programming Notes</p> <p>The following applies to Real Tile Decoding mode: (1) This should be programmed with the starting row address of the next slice segment. If the current slice is last slice of tile (or single slice tile), this should be programmed to the starting row address of the next tile in raster order (tile to the right, except the most right tiles which then will program to the most left tile on the next row). For the very last slice of the frame (last slice of the most bottom right tile [frame bottom-right boundary]), this should be programmed to 0. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be programmed with the starting column address of the next tile in raster order.</p>	Format:	MBZ	Format:	U11
Format:	MBZ				
Format:	U11				

HCP_SLICE_STATE		
3	15	Reserved(for Hardware) Format: MBZ HW uses in Dynamic Slice Mode
	14:10	Reserved Format: MBZ
	9:0	NextSliceStartCtbX or (next_slice_start_lcu_x encoder) Format: U10 Specifies the starting column address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame width (at least +1). <div style="text-align: center; background-color: #e1eef6; padding: 2px;">Programming Notes</div> In VDENC mode, this field should always be programmed as zero The following applies to Real Tile Decoding mode: (1) This should be programmed with the starting column address of the next slice segment. If the current slice is last slice of tile (or single slice tile), this should be programmed to the starting column address of the next tile in raster order (tile to the right, except most right tiles which then will program to the most left tile on the next row). For the very last slice of the frame (last slice of the most bottom right tile [frame bottom-right boundary]), this should be programmed to 0. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be programmed with the starting column address of the next tile in raster order.
3	31:26	Reserved Format: MBZ
	25	LastSliceOfTileColumn Format: U1 Indicates Last Slice of a Tile Column <div style="text-align: center; background-color: #e1eef6; padding: 2px;">Programming Notes</div> The following applies to Real Tile Decoding mode: (1) This should only be set to "1" on the very last slice of every tile column. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should only be set on the most bottom tile/slice on each tile columns.
	24	LastSliceOfTile Format: U1 Indicates last slice of a Tile <div style="text-align: center; background-color: #e1eef6; padding: 2px;">Programming Notes</div> The following applies to Real Tile Decoding mode: (1) This should only be set to "1" on the last slice of every tile.

HCP_SLICE_STATE

	(2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be set on every tiles.																																											
23	cu_chroma_qp_offset_enabled_flag <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This specifies that the cu_chroma_qp_offset_flag may be present in the transform unit syntax. chroma_qp_offset_list_enabled_flag equal to 0 specifies that the cu_chroma_qp_offset_flag is not present in the transform unit syntax. When ChromaArrayType is equal to 0, it is a requirement of bitstream conformance that the value of chroma_qp_offset_list_enabled_flag shall be equal to 0.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th colspan="2" style="text-align: center; background-color: #e1eef6;">Programming Notes</th> </tr> <tr> <td colspan="2" style="padding: 5px;">Decoder only feature.</td> </tr> </table>		Format:	Enable	Programming Notes		Decoder only feature.																																					
Format:	Enable																																											
Programming Notes																																												
Decoder only feature.																																												
22	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ																																								
Format:	MBZ																																											
21:17	slice_cr_qp_offset <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>S4</td> </tr> </table> <p>For deblocking purpose, the pic and slice level cr qp offset must be provided separately. PAK needs to perform $\text{final_chroma_cr_qp_offset} = \text{pic_cr_qp_offset} + \text{slice_cr_qp_offset}$.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center; background-color: #e1eef6;">Value</th> <th style="width: 50%; text-align: center; background-color: #e1eef6;">Name</th> </tr> </thead> <tbody> <tr><td>14h</td><td style="text-align: center;">-12</td></tr> <tr><td>15h</td><td style="text-align: center;">-11</td></tr> <tr><td>16h</td><td style="text-align: center;">-10</td></tr> <tr><td>17h</td><td style="text-align: center;">-9</td></tr> <tr><td>18h</td><td style="text-align: center;">-8</td></tr> <tr><td>19h</td><td style="text-align: center;">-7</td></tr> <tr><td>1Ah</td><td style="text-align: center;">-6</td></tr> <tr><td>1Bh</td><td style="text-align: center;">-5</td></tr> <tr><td>1Ch</td><td style="text-align: center;">-4</td></tr> <tr><td>1Dh</td><td style="text-align: center;">-3</td></tr> <tr><td>1Eh</td><td style="text-align: center;">-2</td></tr> <tr><td>1Fh</td><td style="text-align: center;">-1</td></tr> <tr><td>0h</td><td style="text-align: center;">0</td></tr> <tr><td>1h</td><td style="text-align: center;">1</td></tr> <tr><td>2h</td><td style="text-align: center;">2</td></tr> <tr><td>3h</td><td style="text-align: center;">3</td></tr> <tr><td>4h</td><td style="text-align: center;">4</td></tr> <tr><td>5h</td><td style="text-align: center;">5</td></tr> <tr><td>6h</td><td style="text-align: center;">6</td></tr> </tbody> </table>		Format:	S4	Value	Name	14h	-12	15h	-11	16h	-10	17h	-9	18h	-8	19h	-7	1Ah	-6	1Bh	-5	1Ch	-4	1Dh	-3	1Eh	-2	1Fh	-1	0h	0	1h	1	2h	2	3h	3	4h	4	5h	5	6h	6
Format:	S4																																											
Value	Name																																											
14h	-12																																											
15h	-11																																											
16h	-10																																											
17h	-9																																											
18h	-8																																											
19h	-7																																											
1Ah	-6																																											
1Bh	-5																																											
1Ch	-4																																											
1Dh	-3																																											
1Eh	-2																																											
1Fh	-1																																											
0h	0																																											
1h	1																																											
2h	2																																											
3h	3																																											
4h	4																																											
5h	5																																											
6h	6																																											

HCP_SLICE_STATE

		7h	7
		8h	8
		9h	9
		0Ah	10
		0Bh	11
		0Ch	12
		Programming Notes	
		The valid value is from -12 to 12 (or 14h to 0Ch).	
16:12	slice_cb_qp_offset		
	Format:	S4	
	For deblocking purpose, the pic and slice level cb qp offset must be provided separately.		
	PAK needs to perform $\text{final_chroma_cb_qp_offset} = \text{pic_cb_qp_offset} + \text{slice_cb_qp_offset}$.		
	Value	Name	
	14h	-12	
	15h	-11	
	16h	-10	
	17h	-9	
	18h	-8	
	19h	-7	
	1Ah	-6	
	1Bh	-5	
	1Ch	-4	
	1Dh	-3	
	1Eh	-2	
	1Fh	-1	
	0h	0	
	1h	1	
	2h	2	
	3h	3	
	4h	4	
	5h	5	
	6h	6	
	7h	7	
	8h	8	

HCP_SLICE_STATE									
	<table border="1"> <tr> <td>9h</td> <td>9</td> </tr> <tr> <td>0Ah</td> <td>10</td> </tr> <tr> <td>0Bh</td> <td>11</td> </tr> <tr> <td>0Ch</td> <td>12</td> </tr> </table>	9h	9	0Ah	10	0Bh	11	0Ch	12
9h	9								
0Ah	10								
0Bh	11								
0Ch	12								
	<p style="text-align: center;">Programming Notes</p> <p>The valid value is from -12 to 12 (or 14h to 0Ch).</p>								
11:6	<p>SliceQp</p> <table border="1"> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>Specifies the initial absolute value of QPy quantization parameter for the slice as defined in the Slice Header Semantics section of the HEVC standard. This signifies only the magnitude of SliceQp. In 8 bit, SliceQp only goes from 0 to 51. But in 10 bit, it needs to go from -12 to 51. There is a sign bit specifies at bit [3] below.</p> <p style="text-align: center;">Programming Notes</p> <p>In 12 bit, this needs to go from -24 to 51</p>	Format:	U6						
Format:	U6								
5	<p>slice_temporal_mvp_enable_flag</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>Must be same for all the slices within a frame in encoder mode (follow spec)</p>	Format:	U1						
Format:	U1								
4	<p>dependent_slice_flag</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>Decoder only.</p>	Format:	U1						
Format:	U1								
3	<p>SliceQp Sign Flag</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>This specifies the sign bit of SliceQp. This is added for HEVC 10 bit. For 8 bit, SliceQp goes from 0 to 51 so this bit should be zero. In 10 bit, SliceQp goes from -12 to 51 and this bit can be set for negative value.</p>	Format:	U1						
Format:	U1								
2	<p>LastSliceofPic</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>This indicates the current slice is the very last slice of the current picture</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Not the last slice of the picture</td> </tr> <tr> <td>1</td> <td>Last slice of the picture</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>The following applies to Real Tile Decoding mode:</p>	Format:	U1	Value	Name	0	Not the last slice of the picture	1	Last slice of the picture
Format:	U1								
Value	Name								
0	Not the last slice of the picture								
1	Last slice of the picture								

HCP_SLICE_STATE												
	<p>(1) This should only be set to "1" on the very last slice of most right tile column (frame boundary right-bottom tile).</p> <p>(2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should only be set on the last tile (bottom) of the most right column of the frame.</p>											
1:0	<p>slice_type</p> <p>Format: U2</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>B-slice</td> </tr> <tr> <td>1</td> <td>P-slice</td> </tr> <tr> <td>2</td> <td>I-slice</td> </tr> <tr> <td>3</td> <td>Illegal/Reserved</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, for HEVC standard this field can be 0 or 2 only.</p>	Value	Name	0	B-slice	1	P-slice	2	I-slice	3	Illegal/Reserved	
Value	Name											
0	B-slice											
1	P-slice											
2	I-slice											
3	Illegal/Reserved											
4	<p>31:29 Reserved</p> <p>Format: MBZ</p>											
	<p>28:26 CollocatedRefIDX</p> <p>Format: U3</p> <p>Collocated Motion Vector Temporal Buffer Index.</p> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, the valid values are 0,1,2.</p>											
	<p>25:23 MaxMergeIDX</p> <p>Format: U3</p> <p>MaxNumMergeCand = 5 - five_minus_max_num_merge_cand - 1.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> <tr> <td>2</td> <td>2</td> </tr> <tr> <td>3</td> <td>3</td> </tr> <tr> <td>4</td> <td>4</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>The valid value is from 0 to 4 (MaxNumMergeCand = 5 - five_minus_max_num_merge_cand - 1)</p> <p>In VDENC mode, this field should be programmed as 4.</p>	Value	Name	0	0	1	1	2	2	3	3	4
Value	Name											
0	0											
1	1											
2	2											
3	3											
4	4											
22	<p>cabac_init_flag</p>											

HCP_SLICE_STATE		
	Format:	U1
Programming Notes		
In VDEnc Mode, this value should be the same across all the slices within frame.		
21:19	luma_log2_weight_denom	
	Format:	U3
18:16	ChromaLog2WeightDenom	
	Format:	U3
15	collocated_from_I0_flag	
	Format:	U1
14	isLowDelay	
	Format:	U1
If the POCs of all pictures in both lists are less than the current POC, then set to one, else set to zero.		
Programming Notes		
In VDENC mode, this bit should be set to 1.		
13	mvd_I1_zero_flag	
	Format:	U1
Decoder only.		
12	slice_sao_luma_flag	
	Format:	U1
Programming Notes		
Note: For encoder, all Slices must have same setting within a picture		
11	slice_sao_chroma_flag	
	Format:	U1
Programming Notes		
Note: For encoder, all Slices must have same setting within a picture		
10	slice_loop_filter_across_slices_enabled_flag	
	Format:	U1
Programming Notes		
In VDENC mode, this bit should be set to 1 always.		
No encoder restriction going forward		
9	Reserved	
	Format:	MBZ

HCP_SLICE_STATE						
8:5	slice_beta_offset_div2 or (final Beta_Offset_div2 Encoder) Format: S3 Deblocking filter beta offset. Specified in 2's comp.					
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 70%; text-align: center;">Value</th> <th style="width: 30%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[1101b,0011b]</td> <td style="text-align: center;">[-3,3]</td> </tr> </tbody> </table>	Value	Name	[1101b,0011b]	[-3,3]	
	Value	Name				
	[1101b,0011b]	[-3,3]				
	Programming Notes					
	Note: In VDEnc mode, the value should be the same across all the slices within frame.					
	Valid only in encoder mode					
	4:1	slice_tc_offset_div2 or (final tc_offset_div2 Encoder) Format: S3 Deblocking filter tc offset. Specified in 2's comp.				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 70%; text-align: center;">Value</th> <th style="width: 30%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[1101b,0011b]</td> <td style="text-align: center;">[-3,3]</td> </tr> </tbody> </table>	Value	Name	[1101b,0011b]	[-3,3]
		Value	Name			
[1101b,0011b]		[-3,3]				
Programming Notes						
Note: In VDEnc mode, the value should be the same across all the slices within frame.						
Valid only in encoder mode						
0	slice_header_disable_deblocking_filter_flag Format: U1					
	Programming Notes					
	In VDENC mode, set this bit to zero always.					
5	31:16 Reserved Format: MBZ					
	15:0 SliceHeaderLength Format: U16 Decoder only. Specifies the length in bytes of the slice header including the start code. The starting byte of the slice header in the bit stream buffer is indicated by the Indirect Data Start Address in the HCP_BSD_OBJECT command. The ending byte of the slice header in the same bit stream buffer is indicated by the last byte prior to the slice data (CABAC).					
6	31:30 Reserved Format: MBZ					
	29:26 RoundInter Format: U4 In VDENC mode, this field is ignored.					

HCP_SLICE_STATE																																					
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr><td>0h</td><td>+1/32</td></tr> <tr><td>1h</td><td>+2/32</td></tr> <tr><td>2h</td><td>+3/32</td></tr> <tr><td>3h</td><td>+4/32</td></tr> <tr><td>4h</td><td>+5/32 [Default]</td></tr> <tr><td>5h</td><td>+6/32</td></tr> <tr><td>6h</td><td>+7/32</td></tr> <tr><td>7h</td><td>+8/32</td></tr> <tr><td>8h</td><td>+9/32</td></tr> <tr><td>9h</td><td>+10/32</td></tr> <tr><td>Ah</td><td>+11/32</td></tr> <tr><td>Bh</td><td>+12/32</td></tr> <tr><td>Ch</td><td>+13/32</td></tr> <tr><td>Dh</td><td>+14/32</td></tr> <tr><td>Eh</td><td>+15/32</td></tr> <tr><td>Fh</td><td>+16/32</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Encoder only feature</td> </tr> </tbody> </table>	Value	Name	0h	+1/32	1h	+2/32	2h	+3/32	3h	+4/32	4h	+5/32 [Default]	5h	+6/32	6h	+7/32	7h	+8/32	8h	+9/32	9h	+10/32	Ah	+11/32	Bh	+12/32	Ch	+13/32	Dh	+14/32	Eh	+15/32	Fh	+16/32	Programming Notes	Encoder only feature
Value	Name																																				
0h	+1/32																																				
1h	+2/32																																				
2h	+3/32																																				
3h	+4/32																																				
4h	+5/32 [Default]																																				
5h	+6/32																																				
6h	+7/32																																				
7h	+8/32																																				
8h	+9/32																																				
9h	+10/32																																				
Ah	+11/32																																				
Bh	+12/32																																				
Ch	+13/32																																				
Dh	+14/32																																				
Eh	+15/32																																				
Fh	+16/32																																				
Programming Notes																																					
Encoder only feature																																					
25:24	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																																		
Format:	MBZ																																				
23:20	RoundIntra <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> </table> <p>In VDENC mode, this field is ignored.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr><td>0h</td><td>+1/32</td></tr> <tr><td>1h</td><td>+2/32</td></tr> <tr><td>2h</td><td>+3/32</td></tr> <tr><td>3h</td><td>+4/32</td></tr> <tr><td>4h</td><td>+5/32 [Default]</td></tr> <tr><td>5h</td><td>+6/32</td></tr> <tr><td>6h</td><td>+7/32</td></tr> <tr><td>7h</td><td>+8/32</td></tr> <tr><td>8h</td><td>+9/32</td></tr> <tr><td>9h</td><td>+10/32</td></tr> <tr><td>Ah</td><td>+11/32</td></tr> </tbody> </table>	Format:	U4	Value	Name	0h	+1/32	1h	+2/32	2h	+3/32	3h	+4/32	4h	+5/32 [Default]	5h	+6/32	6h	+7/32	7h	+8/32	8h	+9/32	9h	+10/32	Ah	+11/32										
Format:	U4																																				
Value	Name																																				
0h	+1/32																																				
1h	+2/32																																				
2h	+3/32																																				
3h	+4/32																																				
4h	+5/32 [Default]																																				
5h	+6/32																																				
6h	+7/32																																				
7h	+8/32																																				
8h	+9/32																																				
9h	+10/32																																				
Ah	+11/32																																				

HCP_SLICE_STATE												
		<table border="1"> <tr> <td>Bh</td> <td>+12/32</td> </tr> <tr> <td>Ch</td> <td>+13/32</td> </tr> <tr> <td>Dh</td> <td>+14/32</td> </tr> <tr> <td>Eh</td> <td>+15/32</td> </tr> <tr> <td>Fh</td> <td>+16/32</td> </tr> </table>	Bh	+12/32	Ch	+13/32	Dh	+14/32	Eh	+15/32	Fh	+16/32
Bh	+12/32											
Ch	+13/32											
Dh	+14/32											
Eh	+15/32											
Fh	+16/32											
		Programming Notes										
		Encoder only feature										
	19:0	Reserved Format: MBZ										
7	31:11	Reserved Format: MBZ										
	10	Header Insertion Enable Format: U1 Must be followed by the PAK Insertion Object Command to perform the actual insertion. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>No header insertion into the output bitstream buffer, before the current slice encoded bits.</td> </tr> <tr> <td>1</td> <td></td> <td>Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits.</td> </tr> </tbody> </table>	Value	Name	Description	0		No header insertion into the output bitstream buffer, before the current slice encoded bits.	1		Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits.	
Value	Name	Description										
0		No header insertion into the output bitstream buffer, before the current slice encoded bits.										
1		Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits.										
		Programming Notes										
		Must be always enabled. Encoder Only feature In VDENC mode, this bit should be set to 1.										
	9	SliceData Enable Format: U1 Must always be enabled. Encoder only feature. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>No operation; no insertion.</td> </tr> <tr> <td>1</td> <td></td> <td>Slice Data insertion by PAK Object Commands into the output bitstream buffer.</td> </tr> </tbody> </table>	Value	Name	Description	0		No operation; no insertion.	1		Slice Data insertion by PAK Object Commands into the output bitstream buffer.	
Value	Name	Description										
0		No operation; no insertion.										
1		Slice Data insertion by PAK Object Commands into the output bitstream buffer.										
		Programming Notes										
		In VDENC mode, this bit should be set to 1.										
	8	Tail Insertion Enable Format: U1 Must be followed by the PAK Insertion Object Command to perform the actual insertion.										

HCP_SLICE_STATE											
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>No tail insertion into the output bitstream buffer, after the current slice encoded bits.</td> </tr> <tr> <td>1</td> <td></td> <td>Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits. Tail insertion is only possible at the end of frame but not in the middle (say slice end)</td> </tr> </tbody> </table>	Value	Name	Description	0		No tail insertion into the output bitstream buffer, after the current slice encoded bits.	1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits. Tail insertion is only possible at the end of frame but not in the middle (say slice end)	
Value	Name	Description									
0		No tail insertion into the output bitstream buffer, after the current slice encoded bits.									
1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits. Tail insertion is only possible at the end of frame but not in the middle (say slice end)									
	Programming Notes										
	Tail Insertion is allowed only at the end of last slice or last tile of a frame but not in the middle of frame. Also, no multiple tail insertions are allowed. Encoder only feature										
7:3	Reserved										
	Format:	MBZ									
2	EmulationByteSliceInsertEnable										
	Format:	U1									
	To have PAK outputting SODB or EBSP to the output bitstream buffer.										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>outputting RBSP</td> </tr> <tr> <td>1</td> <td>outputting EBSP</td> </tr> </tbody> </table>	Value	Name	0	outputting RBSP	1	outputting EBSP				
Value	Name										
0	outputting RBSP										
1	outputting EBSP										
	Programming Notes										
	Encoder Only feature										
	In VDENC mode this bit should be set to 1.										
1	CabacZeroWordInsertionEnable										
	Format:	U1									
	To pad the end of a SliceLayer RBSP to meet the encoded size requirement.										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>No Cabac_Zero_Word Insertion.</td> </tr> <tr> <td>1</td> <td></td> <td>Allow internal Cabac_Zero_Word generation and append to the end of RBSP (effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDS).</td> </tr> </tbody> </table>	Value	Name	Description	0		No Cabac_Zero_Word Insertion.	1		Allow internal Cabac_Zero_Word generation and append to the end of RBSP (effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDS).	
Value	Name	Description									
0		No Cabac_Zero_Word Insertion.									
1		Allow internal Cabac_Zero_Word generation and append to the end of RBSP (effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDS).									
	Programming Notes										
	Encoder Only feature										
0	Dependent Slice due to Tile Split										
	Format:	U1									
	In real tile decoding mode or SCC Palette mode, if a single slice covers multiple tiles, it is required for SW to split the slice into multiple slices per tile for HW programming. This bit should										

HCP_SLICE_STATE															
	<p>be set on all slices split out from original slice (except the first slice split out).</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">This is not used by HW. This is used for validation only.</td> </tr> </table>	Programming Notes		This is not used by HW. This is used for validation only.											
Programming Notes															
This is not used by HW. This is used for validation only.															
8	31:29 Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ												
	Format:	MBZ													
	28:6 Indirect PAK-BSE Data Start Offset (Write) <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U23</td> </tr> </table> <p>This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the HCP PAK-BSE Object Base Address.</p> <p>It is a cacheline-aligned address for the HEVC bitstream data.</p> <p>For Write, there is no need to have a data length field. It is assumed the global memory upper bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,524288]</td> <td></td> </tr> </tbody> </table> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Must be zero.</td> </tr> <tr> <td colspan="2">Encoder Only feature</td> </tr> <tr> <td colspan="2">In VDENC mode, this field should be zero.</td> </tr> </table>	Format:	U23	Value	Name	[0,524288]		Programming Notes		Must be zero.		Encoder Only feature		In VDENC mode, this field should be zero.	
	Format:	U23													
Value	Name														
[0,524288]															
Programming Notes															
Must be zero.															
Encoder Only feature															
In VDENC mode, this field should be zero.															
5:0 Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ														
9	31 Force SAO parameters to zero 1->Force SAO parameters to zero 0->normal compute This bit does not change SAO compute but forces SAO parameters to zero at the end. Default should be 0. All slices must be programmed to same value within a frame. This bit is used for rate control purpose on last pass along with delta QP to decrease bitstream size. All previous passes still use delta QP to decrease/increase bitstream size. HW might not hit the last pass. <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only feature</td> </tr> </table>	Programming Notes		Encoder Only feature											
	Programming Notes														
	Encoder Only feature														
30:16 Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ														
15:0 Transformskip_lambda Lambda value used in transform skip calculation <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </table>	Programming Notes														
Programming Notes															

HCP_SLICE_STATE		
		Encoder Only feature
10	31:24	Transformskip_numnonzerocoeffs_factor1 Multiplying factor with number of non-zero coefficients for non-Skip pipe calculation <div style="text-align: center;">Programming Notes</div> Encoder Only feature
	23:16	Transformskip_numzerocoeffs_factor1 Multiplying factor with number of zero coefficients for non-Skip pipe calculation <div style="text-align: center;">Programming Notes</div> Encoder Only feature
	15:8	Transformskip_numnonzerocoeffs_factor0 Multiplying factor with number of non-zero coefficients for skip pipe calculation <div style="text-align: center;">Programming Notes</div> Encoder Only
	7:0	Transformskip_numzerocoeffs_factor0 Multiplying factor with number of zero coefficients for skip pipe calculation <div style="text-align: center;">Programming Notes</div> Encoder Only feature
11	31:26	Reserved Format: MBZ
	25:16	OriginalSliceStartCtbY Format: U10 This field programs the Slice CTB horizontal starting position of the full slice (independent and dependent slices are counted as single full slice).. If this slice is an independent slice and (if slice spans multiple tiles) first tile of this independent slice, this field should have the same value as SliceStartCtbY (above). Otherwise, this field should have the original SliceStartCtbY of the real slice. This field is only used in decoder scalable mode
	15:10	Reserved Format: MBZ
	9:0	OriginalSliceStartCtbX Format: U10 This field programs the Slice CTB vertical starting position of the full slice (independent and dependent slices are counted as single full slice). If this slice is an independent slice and (if slice spans multiple tiles) first tile of this independent slice, this field should have the same value as SliceStartCtbX (above). Otherwise, this field should have the original SliceStartCtbX of the actual slice. This field is only used in decoder scalable mode
12	31	use_integer_mv_flag This specifies that the resolution of motion vectors for inter prediction in the current slice is integer.

HCP_SLICE_STATE			
	<p>When not present, the value of use_integer_mv_flag is inferred to be equal to motion_vector_resolution_control_idc. Decoder only [Encoder default to 0]</p>		
30:18	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
17:12	<p>slice_act_y_qp_offset This specifies offsets to the quantization parameter values qP derived in clause 8.6.2 for luma components. The values of slice_act_cb_y_offset shall be in the range of 12 to +12, inclusive. The value of PpsActQpOffsetY+ slice_act_y_qp_offset shall be in the range of 12 to +12, inclusive. This applies to both decoder and encoder.</p>		
11:6	<p>slice_act_cb_qp_offset This specifies offsets to the quantization parameter values qP derived in clause 8.6.2 for Cb components. The values of slice_act_cb_qp_offset shall be in the range of 12 to +12, inclusive. The value of PpsActQpOffsetCb+ slice_act_cb_qp_offset shall be in the range of 12 to +12, inclusive. This applies to both decoder and encoder.</p>		
5:0	<p>slice_act_cr_qp_offset This specifies offsets to the quantization parameter values qP derived in clause 8.6.2 for Cr components. The values of slice_act_cr_qp_offset shall be in the range of 12 to +12, inclusive. The value of PpsActQpOffsetCr+ slice_act_cr_qp_offset shall be in the range of 12 to +12, inclusive. This applies to both decoder and encoder.</p>		

HCP_SURFACE_STATE

HCP_SURFACE_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>The HCP_SURFACE_STATE command is responsible for defining the frame buffer pitch and the offset of the chroma component.</p> <p>This is a picture level state command and is shared by both encoding and decoding processes.</p> <p>Note : When NV12/P010 and Tile Y are being used, full pitch and interleaved UV is always in use. U and V X offset must be set to 0; U and V Y offset must be 4-pixel aligned. For 10-bit pixel, P010 surface definition is being used.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = HCP = 7h	
22:16	Media Instruction Command		
	Default Value:	1h HCP_SURFACE_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	1h		
1	31:28	Surface Id	
		Format:	U4

HCP_SURFACE_STATE			
	Value	Name	Description
	0h	HEVC: For current decoded Picture	8-bit uncompressed data
	1h	Source Input Picture (encoder)	8-bit uncompressed data
	2h	Prev Reference Picture	(VP9 only) Previous Reference
	3h	Golden Reference Picture	(VP9 only) Golden Reference
	4h	AltRef Reference Picture	(VP9 only) AltRef Reference
	5h	HEVC: Reference Pictures	(HEVC only) Reference. Also, this will have separate compressible bits per reference surfaces for HEVC
27:17	Reserved		
	Format:	MBZ	
16:0	Surface Pitch Minus1		
	Format:	U17-1	
	This field specifies the surface pitch in (#Bytes - 1).		
	Programming Notes		
	For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case). The range in bytes is $[2^{Cu}-1, 131071]$ -> $[(2^{Cu})B, 128KB] = [1 \text{ tile}, 128KB/(2^{Cu} \text{ tiles})]$		
	The field specifies the surface pitch in (#Bytes - 1)		
	For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]		
2	31:27	Surface Format	
	Format:	U5	
	Specifies the format of the surface.		
	Value	Name	Description
	0h	YUY2 format	
	1h	RGB_8 format	
	2h	AYUV4444 format	
	3h	P010Variant	P010Variant is a modified P010 format, >8 bit planar 420 with MSB together and LSB at an offset in x direction where the x-offset should be 32-bit

HCP_SURFACE_STATE			
			aligned.
4h	PLANAR_420_8		
5h	YCRCB_SwapY format		
6h	YCRCB_SwapUV format		
7h	YCRCB_SwapUVY format		
8h	Y216/Y210 format	Same value is used to represent Y216 and Y210	
9h	RGB_10 format		
Ah	Y410 format		
Bh	NV21 Planar_420_8 Format		
Ch	Y416 format		
Dh	P010		
11h	Y216Variant	Y216Variant is the modified Y210/Y216 format, 8 bit planar 422 with MSB bytes packed together and LSB bytes at an offset in the X-direction where the x-offset is 32-bit aligned. The chroma is UV interleaved with identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.	
12h	Y416Variant	Y416Variant is the modified Y410/Y412/Y416 format, 8 bit planar 444 with MSB bytes packed together and LSB bytes at an offset in the X-direction where the x-offset is 32-bit aligned. The U channel is below the luma, has identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma. The V channel is below the U, has identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.	
13h	YUY2Variant	YUY2Variant is the modified YUY2 format, 8 bit planar 422. The chroma is UV interleaved and is at an offset in the Y-direction (similar to NV12) but is the same	

HCP_SURFACE_STATE			
			height as the luma.
	14h	AYUV4444Variant	AYUV4444Variant is the modified AYUV4444 format, 8 bit planar 444 format. The U channel is below the luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma. The V channel is below the and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.
	15h-1Fh	Reserved	
Programming Notes			
Programming restriction on HEVC decoder: <ol style="list-style-type: none"> If both luma_bitdepth_minus8 and chroma_bitdepth_minus 8 are both 0 (8 bits for both luma/chroma), this should be programmed to PLANAR_420_8 If either luma_bitdepth_minus8 or chroma_bitdepth_minus 8 is non-zero (9 or 10 bits for either or both luma/chroma), this should be programmed to P010. 			
26	Reserved		
	Format:	MBZ	
25	Reserved		
	Format:	MBZ	
24:15	Reserved		
	Format:	MBZ	
14:0	Y Offset for U(Cb) in pixel		
	Format:	U15	
This field specifies the vertical offset in rows from the Surface Base Address to the start(origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats.			
Programming Notes			
<ul style="list-style-type: none"> For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the Memory Address Attributes table. TileY (legacy 4k) - 8 pixel aligned TileYF (New 4k) - 64 pixel aligned TileYS (64k) - 256 pixel aligned 			
3	31:16	Reserved	
	Format:	MBZ	
	15:0	Default Alpha Value	

HCP_SURFACE_STATE									
4	31:21	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ					
	Format:	MBZ							
	20:16	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ					
	Format:	MBZ							
15:8	Compression Type This field indicates if the compression type for the reference surface is media or render compressed. In HEVC mode, each bit is used for 1 reference starting with Bit 8 for Ref 0 in the ref list and Bit 9 for Ref 1 and so on. In VP9 mode, Bit 8 is for Previous Reference; Bit 9 is for Golden Reference and Bit 10 is for Alternate Reference; Bits 11-15 are unused and should be programmed to 0 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Media compression Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; color: #0070C0;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.</td> </tr> </tbody> </table>	Value	Name	0	Media compression Enabled [Default]	1	Render Compression Enabled	Programming Notes	In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.
Value	Name								
0	Media compression Enabled [Default]								
1	Render Compression Enabled								
Programming Notes									
In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.									
7:0	Memory Compression Enable In HEVC mode, each bit is used for 1 reference starting with Bit 0 for Ref 0 in the ref list and Bit 1 for Ref 1 and so on. In VP9 mode, Bit 0 is for Previous Reference; Bit 1 is for Golden Reference and Bit 2 is for Alternate Reference; Bits 3-7 are unused and should be programmed to 0. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Memory Compression Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Memory Compression Disable</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; color: #0070C0;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.</td> </tr> </tbody> </table>	Value	Name	1	Memory Compression Enable	0	Memory Compression Disable	Programming Notes	In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.
Value	Name								
1	Memory Compression Enable								
0	Memory Compression Disable								
Programming Notes									
In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.									

HCP_TILE_CODING

HCP_TILE_CODING				
Source:	BSpec			
Length Bias:	2			
Programming Notes				
This command is used for both HEVC and VP9 codecs				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	Opcode	
	28:27	28:27	Pipeline Type	
			Default Value:	2h
			Format:	Opcode
	26:23	26:23	Media Instruction Opcode	
			Default Value:	7h Codec/Engine Name
			Format:	Opcode
	22:16	22:16	Media Instruction Command	
Default Value:			15h HCP_TILE_CODING	
Format:			Opcode	
15:12	15:12	Reserved		
		Format:	MBZ	
11:0	11:0	Dword Length		
		Format:	=n	
		Description		
		Excludes Dwords 0 & 1		
1	31:16	Num of Tile columns in a Frame		
		Description		
		Specify the total number of Tile columns in a frame.		
		This field is not used by HW.		
		Restriction : This field should be same as "Number of Active BE Pipes" defined in this register for the following modes:		
		(1) HEVC/VP9 virtual tile decode mode		
		(2) HEVC/VP9 PAK (encoder) mode		
		In HEVC decoder real tile scalability, this field should indicate the total number of tile columns in the actual frame bitstream. The field should be larger than or equal to "Number of Active BE Pipes"		

HCP_TILE_CODING

	15:10	Reserved MBZ																																							
	9	<p>Tile Column store Select This bit is used for computing Tile Column store write offset and Tile read column store read address.</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p style="text-align: center; color: blue; font-weight: bold;">Programming Notes</p> <p>Tile Configuration 1: 3x3 tiles</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table> <p>In the above tiling configuration, For Tiles 1,4,7,3,6,9 Tile Columnstore select should be programmed as zero. For Tiles 2,5,8 Tile Row store select should be programmed as 1.</p> <p>Tile Configuration 1: 3x5 tiles</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr> <tr><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr> </table> <p>In the above tiling configuration, For Tiles 1,6,11, 3,8, 13, 5, 10, 15 Tile Columnstore select should be programmed as zero. For Tiles 2,7, 12, 4, 9, 14 Tile Columnstore select should be programmed as 1.</p> <p>Tile Configuration 1: 3x5 tiles</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> <tr><td>10</td><td>11</td><td>12</td></tr> <tr><td>13</td><td>14</td><td>15</td></tr> </table> <p>In the above tiling configuration, For Tiles 1,4,7,10,13, 3,6, 9, 12, 15 Tile Columnstore select should be programmed as zero. For Tiles 2,5,8,11,14 Tile Columnstore select should be programmed as 1.</p> </div>	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3																																							
4	5	6																																							
7	8	9																																							
1	2	3	4	5																																					
6	7	8	9	10																																					
11	12	13	14	15																																					
1	2	3																																							
4	5	6																																							
7	8	9																																							
10	11	12																																							
13	14	15																																							
	8	<p>Tile Row store Select This bit is used for computing Tile row store write offset and Tile read row store read address.</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p style="text-align: center; color: blue; font-weight: bold;">Programming Notes</p> <p>Tile Configuration 1: 3x3 tiles</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table> <p>In the above tiling configuration, For Tiles 1,2,3,7,8,9 Tile Row store select should be programmed as zero. For Tiles 4,5,6 Tile Row store select should be programmed as 1.</p> <p>Tile Configuration 1: 3x5 tiles</p> </div>	1	2	3	4	5	6	7	8	9																														
1	2	3																																							
4	5	6																																							
7	8	9																																							

HCP_TILE_CODING

		<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr> <tr><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr> </table>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	<p>In the above tiling configuration, For Tiles 1,2,3,4,5,11,12,13,14,15 Tile Row store select should be programmed as zero. For Tiles 6,7,8,9,10 Tile Row store select should be programmed as 1.</p> <p>Tile Configuration 1: 3x5tiles</p> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> <tr><td>10</td><td>11</td><td>12</td></tr> <tr><td>13</td><td>14</td><td>15</td></tr> </table> <p>In the above tiling configuration, For Tiles 1,2,3,7,8,9,13,14,15 Tile Row store select should be programmed as zero. For Tiles 4,5,6,10,11,12 Tile Row store select should be programmed as 1.</p>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5																													
6	7	8	9	10																													
11	12	13	14	15																													
1	2	3																															
4	5	6																															
7	8	9																															
10	11	12																															
13	14	15																															
	7:0	<p>Number of Active BE Pipes</p> <p>Indicates the number of active, consecutive positioned Scalable VDBOXs to be used for the current frame decoding or encoding. BE Pipe partitioning, SW must guarantee the minimum width is at least two full LCUs for each tiles</p> <p>This field in general should be smaller or equal to Num of Tile columns in a Frame. This field is ignored by HW</p> <p>This field is not used by HW</p> <table border="1" style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Comment</th> </tr> </thead> <tbody> <tr><td>0</td><td>ignored</td></tr> <tr><td>1</td><td>ignored</td></tr> <tr><td>2</td><td>Supported by Encoder / Decoder.</td></tr> <tr><td>3</td><td>Supported only by Decoder.</td></tr> <tr><td>4</td><td>Supported only by Encoder</td></tr> </tbody> </table>		Value	Comment	0	ignored	1	ignored	2	Supported by Encoder / Decoder.	3	Supported only by Decoder.	4	Supported only by Encoder																		
Value	Comment																																
0	ignored																																
1	ignored																																
2	Supported by Encoder / Decoder.																																
3	Supported only by Decoder.																																
4	Supported only by Encoder																																
2	31	<p>IsLastTileOfColumn</p> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Indicates if current Tile is last tile of a Column</p>		Format:	U1																												
Format:	U1																																
	30	<p>IsLastTileOfRow</p> <p>Indicates if current Tile is Last Tile of a Row</p>																															
	29:26	<p>Reserved</p> <table border="1" style="border-collapse: collapse; width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>		Format:	MBZ																												
Format:	MBZ																																

HCP_TILE_CODING				
	25:16	Tile Row Position <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U10</td> </tr> </table> Ctb row position of tile For VP9: In units of SB64x64	Format:	U10
	Format:	U10		
	15:11	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
10	Non First Pass Tile Indicates Non first past Tile when Tile Replay Mode is enabled			
9:0	Tile Column Position <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U10</td> </tr> </table> Ctb column position of tile For VP9: In units of SB64x64	Format:	U10	
Format:	U10			
3	31	LastPassOfTile (ValidationOnly) This bit indicates last pass of a Tile. This is validation use only. HW/SW should not use in design		
	30:27	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	26:16	TileWidthInMinCbMinus1 Specifies Tile width in units of minimum coding block size. The minimal width per tile is at least two full LCUs. In HEVC Encoder mode, the following restrictions apply. Last LCU at frames right edge must align to CU boundary. This applies to all size of LCUs: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. For VP9: In units of 8x8		
	15:11	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
10:0	TileHeightInMinCbMinus1 Specifies Tile Height in units of minimum coding block size In HEVC Encoder mode, the following restrictions apply. Last LCU at frames bottom edge must align to CU boundary. This applies to all size of LCUs: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. For VP9: In units of 8x8			
4	31:6	Bitstream Byte Offset Offset on top of base address from where the encoded bitstream should be written out for this tile In scalability mode: this offset is valid for every tile and it must be zero for the first tile in a frame. Non scalability mode: not valid		

HCP_TILE_CODING

	5:1	Reserved
		Format: MBZ
	0	Bitstream Byte Offset Enable 0-> Disables bitstream byte offset, meaning the encoded bitstream for all TILES would be contiguous This bit must be zero when FPAK mode enabled in non-scalability mode. Note: In FPAK mode, only single Tile Column (1xN) supported in non-scalability mode
5	31:6	PAK Frame Statistics Offset The frame statistics (SSE, RhoDomain and LCU stats) will be reported per Tile. Valid only in scalability mode
	5:0	Reserved Format: MBZ
6	31:6	CU Level Streamout Offset CU level statistics (see details in streamout section) per tile will be streamed out starting from this offset address This offset is valid for every Tile in scalability mode only.
	5:0	Reserved Format: MBZ
7	31:6	Slice Size Streamout Offset Size of every slice within this tile will be streamed out starting from this offset address. This offset is valid for every Tile in scalability mode only.
	5:0	Reserved Format: MBZ
8	31:6	CU record offset Offset address for CU record for this tile This offset is valid for every Tile in scalability mode only.
	5:0	Reserved Format: MBZ
9	31:6	SSE RowStore offset SSE(Sum Square Error) statistics per tile will be written out at this address This offset is valid for every Tile in scalability mode.
	5:0	Reserved Format: MBZ
10	31:6	SAO RowStore offset SAO Rowstore offset for this tile. This offset is valid for every Tile in scalability mode only.
	5:0	Reserved Format: MBZ
11	31:6	Tile Size StreamOut Offset

HCP_TILE_CODING					
		<p>Tile Size will be written out at this offset This offset is valid for every Tile in scalability mode only.</p>			
	5:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
12	31:6	<p>VP9 Probability Counter Streamout Offset Probability counters will be written out starting from this offset address This offset is valid for every Tile in scalability mode only.</p>			
	5:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
13..14	63:0	<p>HCP Scalability Synchronize Buffer - Base Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">SplitBaseAddress64ByteAligned</td> </tr> </table> <p>Specifies the 64 byte aligned buffer address used for data synchronization between neighboring pipes in scalable modes. The buffer will be written and read (as a flush mechanism) by hardware to guarantee data made it to memory before neighboring pipe can read the data. Hardware will also write the current row (in LCU) number to indicate which the current processing rows.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table> <p>This minimal buffer size (in CLs) should be set to the number of scalable pipes used by this workload. This data should be not be cached.</p>	Format:	SplitBaseAddress64ByteAligned	Programming Notes
Format:	SplitBaseAddress64ByteAligned				
Programming Notes					
15	31:0	<p>HCP Scalability Synchronize Buffer - Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes	
Format:	MemoryAddressAttributes				
16	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
17	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
18	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
19	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				

HCP_TILE_STATE

HCP_TILE_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>This command is valid for decoder only.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
Format:		OpCode	
Codec/Engine Name = HCP = 7h			
22:16	Media Instruction Command		
	Default Value:	11h HCP_TILE_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	Fh		
1	31:10	Reserved	
		Format:	MBZ
	9:5	NumTileColumnsMinus1	
Format:	U5		
Specifies the number of tile columns in Ctbs per picture. Maximum of 20 columns are supported (level 6.2 restriction)			
4:0	NumTileRowsMinus1		

HCP_TILE_STATE				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U5</td> </tr> </table> <p>Specifies the number of tile rows in Ctbs per picture. Maximum of 22 rows are supported (level 6.2 restriction)</p>	Format:	U5
Format:	U5			
2..6	159:0	<p>Ctb column position of tile column</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">HCP_TILE_POSITION_IN_CTB[5]</td> </tr> </table>	Format:	HCP_TILE_POSITION_IN_CTB[5]
Format:	HCP_TILE_POSITION_IN_CTB[5]			
7..12	191:0	<p>Ctb row position of tile row</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">HCP_TILE_POSITION_IN_CTB[6]</td> </tr> </table> <p>Note that there are only 22 rows, so the most significant 16 bits of HCP_TILE_POSITION_IN_CTB[5] (31:16) are reserved</p>	Format:	HCP_TILE_POSITION_IN_CTB[6]
Format:	HCP_TILE_POSITION_IN_CTB[6]			
13..14	63:0	<p>Ctb column position MSB</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">HCP_TILE_POSITION_IN_CTB_MSB</td> </tr> </table>	Format:	HCP_TILE_POSITION_IN_CTB_MSB
Format:	HCP_TILE_POSITION_IN_CTB_MSB			
15..16	63:0	<p>Ctb row position MSB</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">HCP_TILE_POSITION_IN_CTB_MSB</td> </tr> </table>	Format:	HCP_TILE_POSITION_IN_CTB_MSB
Format:	HCP_TILE_POSITION_IN_CTB_MSB			

HCP_VP9_PAK_OBJECT

HCP_VP9_PAK_OBJECT			
Source:		VideoCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = HUC = Bh	
22:16	Media Instruction Command		
	Default Value:	35h HCP_VP9_PAK_OBJECT	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	0h		
1	31	IsLastSB Flag (of Tile)	
		Format:	Enable
		Value	Name
		0	False
	1	True	
	30	Reserved	
		Format:	MBZ
	29:24	CU_count_minus1	
		Format:	U6
		Number of CUs in the current SB = CU_count_minus1 + 1. Minimum, there must be 1 CU in a SB.	

HCP_VP9_PAK_OBJECT														
	23:21	Reserved Format: MBZ												
	20:0	split_coding_unit_flag[x0][y0] Format: U21 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">Bit 20</td> <td>Split_flag_level0</td> </tr> <tr> <td>Bit 19:16</td> <td>Split_flag_level1 [19:16] is in raster order. Bit16 is for partition0 in raster order.</td> </tr> <tr> <td>Bit 15:12</td> <td>Split_flag_level2_level1part3 Split flags for bit19 partition. [15:12] is in raster order. Bit12 is for partition0 in raster order.</td> </tr> <tr> <td>Bit 11:8</td> <td>Split_flag_level2_level1part2 Split flags for bit18 partition. [11:8] is in raster order. Bit8 is for partition0 in raster order.</td> </tr> <tr> <td>Bit 7:4</td> <td>Split_flag_level2_level1part1 Split flags for bit17 partition. [7:4] is in raster order. Bit4 is for partition0 in raster order.</td> </tr> <tr> <td>Bit 3:0</td> <td>Split_flag_level2_level1part0 Split flags for bit16 partition. [3:0] is in raster order. Bit0 is for partition0 in raster order.</td> </tr> </table>	Bit 20	Split_flag_level0	Bit 19:16	Split_flag_level1 [19:16] is in raster order. Bit16 is for partition0 in raster order.	Bit 15:12	Split_flag_level2_level1part3 Split flags for bit19 partition. [15:12] is in raster order. Bit12 is for partition0 in raster order.	Bit 11:8	Split_flag_level2_level1part2 Split flags for bit18 partition. [11:8] is in raster order. Bit8 is for partition0 in raster order.	Bit 7:4	Split_flag_level2_level1part1 Split flags for bit17 partition. [7:4] is in raster order. Bit4 is for partition0 in raster order.	Bit 3:0	Split_flag_level2_level1part0 Split flags for bit16 partition. [3:0] is in raster order. Bit0 is for partition0 in raster order.
Bit 20	Split_flag_level0													
Bit 19:16	Split_flag_level1 [19:16] is in raster order. Bit16 is for partition0 in raster order.													
Bit 15:12	Split_flag_level2_level1part3 Split flags for bit19 partition. [15:12] is in raster order. Bit12 is for partition0 in raster order.													
Bit 11:8	Split_flag_level2_level1part2 Split flags for bit18 partition. [11:8] is in raster order. Bit8 is for partition0 in raster order.													
Bit 7:4	Split_flag_level2_level1part1 Split flags for bit17 partition. [7:4] is in raster order. Bit4 is for partition0 in raster order.													
Bit 3:0	Split_flag_level2_level1part0 Split flags for bit16 partition. [3:0] is in raster order. Bit0 is for partition0 in raster order.													
2	31:16	Current SB Y Addr) Format: U16												
	15:0	Current SB X Addr Format: U16												
3	31:0	Reserved Format: MBZ												
4	31:17	Reserved Format: MBZ												
	16	LcuForceZeroCoeff (Time Budget Overflow Occurred) Format: Enable <p>The bitstream packer pipeline will force all the coefficients to zero if this bit is set. Also, VDenc Indicates to PAK that time budget overflow has occurred.</p>												
	15:12	SSE ClassID 32x32_3 Format: U4 <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <tr> <th colspan="2" style="text-align: center; background-color: #e6f2ff;">Programming Notes</th> </tr> <tr> <td colspan="2">This parameter indicates the SSE classID for the 32x32 block3 of the current LCU. Valid values: 0-8</td> </tr> </table>	Programming Notes		This parameter indicates the SSE classID for the 32x32 block3 of the current LCU. Valid values: 0-8									
Programming Notes														
This parameter indicates the SSE classID for the 32x32 block3 of the current LCU. Valid values: 0-8														
	11:8	SSE ClassID 32x32_2 Format: U4												

HCP_VP9_PAK_OBJECT									
	<table border="1"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">This parameter indicates the SSE classID for the 32x32 block2 of the current LCU. Valid values: 0-8</td> </tr> </table>	Programming Notes		This parameter indicates the SSE classID for the 32x32 block2 of the current LCU. Valid values: 0-8					
Programming Notes									
This parameter indicates the SSE classID for the 32x32 block2 of the current LCU. Valid values: 0-8									
7:4	<table border="1"> <tr> <td colspan="2">SSE ClassID 32x32_1</td> </tr> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8</td> </tr> </table>	SSE ClassID 32x32_1		Format:	U4	Programming Notes		This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8	
SSE ClassID 32x32_1									
Format:	U4								
Programming Notes									
This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8									
3:0	<table border="1"> <tr> <td colspan="2">SSE ClassID 32x32_0</td> </tr> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8</td> </tr> </table>	SSE ClassID 32x32_0		Format:	U4	Programming Notes		This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8	
SSE ClassID 32x32_0									
Format:	U4								
Programming Notes									
This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8									

HCP_VP9_PIC_STATE

HCP_VP9_PIC_STATE			
Source:	VideoCS		
Length Bias:	2		
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
Default Value:		7h Codec/Engine Name	
Format:		OpCode	
Codec/Engine Name = HUC = Bh			
22:16	Media Instruction Command		
	Default Value:	30h HCP_VP9_PIC_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	Programming Notes
	Bh	Decoder DW Length	Only Up to DW12 should be programmed for decoder
1Eh	Encoder DW Length	All DWs should be programmed for encoder	
1	31:30	Reserved	
		Format:	MBZ
	29:16	Frame Height In Pixels Minus 1	
Format:	U14		
Specifies the height of the decoded picture in units of 8 pixels, which is the minimum coding block size. The decoded picture height in units of luma samples equals $(\text{FrameHeightInMinBlocksMinus1} + 1) * 8$ <i>For Encoder Partial SB:</i>			

HCP_VP9_PIC_STATE																			
	<p><i>Kernel, on last SB (at picture edges), splits the SB into as small as possible CUs to have picture boundaries aligned to CU boundary.</i></p> <p><i>Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively.</i></p> <p><i>Driver sets up a SB aligned (both in X/Y direction) surface.</i></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 60%;">Value</th> <th style="width: 40%;">Name</th> </tr> </thead> <tbody> <tr> <td>[4096,16383]</td> <td>4K_TO_16K</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Decoder supports 16K image and 8K video. Encoder only supports up to 8K.</td> </tr> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Description</th> </tr> <tr> <td>0-6</td> <td>Invalid (multiple of 8 pixels)</td> </tr> <tr> <td>7-8191</td> <td>8-8K Pixels (Decoder and Encoder)</td> </tr> <tr> <td>8192-16383</td> <td>8K-16K Pixels (Decoder only)</td> </tr> </tbody> </table>	Value	Name	[4096,16383]	4K_TO_16K	Programming Notes		Decoder supports 16K image and 8K video. Encoder only supports up to 8K.		Value	Description	0-6	Invalid (multiple of 8 pixels)	7-8191	8-8K Pixels (Decoder and Encoder)	8192-16383	8K-16K Pixels (Decoder only)		
Value	Name																		
[4096,16383]	4K_TO_16K																		
Programming Notes																			
Decoder supports 16K image and 8K video. Encoder only supports up to 8K.																			
Value	Description																		
0-6	Invalid (multiple of 8 pixels)																		
7-8191	8-8K Pixels (Decoder and Encoder)																		
8192-16383	8K-16K Pixels (Decoder only)																		
15:14	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																
Format:	MBZ																		
13:0	<p>Frame Width In Pixels Minus 1</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>U14</td> </tr> </table> <p>Specifies the width of the decoded picture in units of minimum coding block size. The decoded picture width in units of luma samples equals $(\text{FrameWidthInMinBlocksMinus1} + 1) * 8 - 1$. This should be programmed to a multiple of 8 pixels minus 1.</p> <p><i>For Encoder Partial SB:</i></p> <p><i>Kernel, on last SB (at picture edges), splits the SB into as small as possible CUs to have picture boundaries aligned to CU boundary.</i></p> <p><i>Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively.</i></p> <p><i>Driver sets up a SB aligned (both in X/Y direction) surface.</i></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 60%;">Value</th> <th style="width: 40%;">Name</th> </tr> </thead> <tbody> <tr> <td>[4096,16383]</td> <td>4K_TO_16K</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Decoder supports 16K image and 8K video. Encoder only supports up to 8K.</td> </tr> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Description</th> </tr> <tr> <td>0-6</td> <td>Invalid (multiple of 8 pixels)</td> </tr> <tr> <td>7-8191</td> <td>8-8K Pixels (Decoder and Encoder)</td> </tr> <tr> <td>8192-16383</td> <td>8K-16K Pixels (Decoder only)</td> </tr> </tbody> </table>	Format:	U14	Value	Name	[4096,16383]	4K_TO_16K	Programming Notes		Decoder supports 16K image and 8K video. Encoder only supports up to 8K.		Value	Description	0-6	Invalid (multiple of 8 pixels)	7-8191	8-8K Pixels (Decoder and Encoder)	8192-16383	8K-16K Pixels (Decoder only)
Format:	U14																		
Value	Name																		
[4096,16383]	4K_TO_16K																		
Programming Notes																			
Decoder supports 16K image and 8K video. Encoder only supports up to 8K.																			
Value	Description																		
0-6	Invalid (multiple of 8 pixels)																		
7-8191	8-8K Pixels (Decoder and Encoder)																		
8192-16383	8K-16K Pixels (Decoder only)																		
2	<p>31 Segment ID StreamIn Enable</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Enable</td> </tr> </table> <p>Indicates SegmentID from previous frame needs to be streamIn for Segment ID</p>	Format:	Enable																
Format:	Enable																		

HCP_VP9_PIC_STATE

prediction									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Disable</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Enable</td> </tr> </tbody> </table>	Value	Name	0	Disable	1	Enable		
Value	Name								
0	Disable								
1	Enable								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; color: #0070c0;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;"> Decoder Only: SegmentIDStreamInEnable = error_resilient_mode OR intra_only OR VP9_KEY_FRAME </td> </tr> </tbody> </table>	Programming Notes	Decoder Only: SegmentIDStreamInEnable = error_resilient_mode OR intra_only OR VP9_KEY_FRAME						
Programming Notes									
Decoder Only: SegmentIDStreamInEnable = error_resilient_mode OR intra_only OR VP9_KEY_FRAME									
30	<p>Segment ID StreamOut Enable</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">Enable</td> </tr> </table> <p>Indicates SegmentID of current frame needs to be streamOut for next frame</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Disable</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Enable</td> </tr> </tbody> </table> <p style="text-align: center; color: #0070c0; margin-top: 10px;">Programming Notes</p> <p>Decoder Only: SegmentIDStreamOutEnable = segmentation_enabled AND segmentation_update_map</p>	Format:	Enable	Value	Name	0	Disable	1	Enable
Format:	Enable								
Value	Name								
0	Disable								
1	Enable								
29	<p>Lossless Mode</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">Enable</td> </tr> </table> <p style="margin-top: 10px;">Set this bit to indicate lossless coding mode.</p> <p>In encoder mode, software has to set tx_mode to 4x4only and all tu_sizes in CU record as 4x4 for entire frame. Software also has to program such that final_qindex=0 and final_filter_level=0 following the Quant Scale and Filter Level Table in Segmentation State section. Hardware forces Hadamard Tx when this bit is set. When Lossless Mode is on, BRC has to be off.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Normal Mode</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Lossless Mode</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	0	Normal Mode	1	Lossless Mode
Format:	Enable								
Value	Name								
0	Normal Mode								
1	Lossless Mode								
28	<p>Segmentation Temporal Update</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">Enable</td> </tr> </table> <p style="margin-top: 10px;">Indicates whether segID is decoding from bitstream or predicted from previous frame.</p> <p>In encoder Mode it should use either from previous frame or streamIn</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 20%; text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">Decode segID from bitstream</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	0h	Decode segID from bitstream		
Format:	Enable								
Value	Name								
0h	Decode segID from bitstream								

HCP_VP9_PIC_STATE

		1h	Get segID either from bitstream or from previous frame
Programming Notes			
Decoder Only: For KEY_FRAME or INTRA_ONLY frame, this bit should be set to "0".			
Note: Driver should override this flag to "0" in KEY_FRAME or INTRA_ONLY frame even if this bit decoded from bitstream is different. This is for hardware optimization. This override does not affect bitstream decoding other than uncompressed header.			
27	Segmentation Update Map		
	Format:	Enable	
Indicates how hardware determines segmentation ID			
	Value	Name	Description
	0h		Intra block: segment ID is zero Inter block: get segment ID from previous frame (streamIN)
	1h		Intra block: decode segment ID from bitstream. Inter block: determines from segmentation_temporal_update setting
26	Segmentation Enabled		
	Format:	Enable	
Indicate if segmentation is enabled or not			
	Value	Name	
	0h	All blocks are implied to belong to segment 0	
	1h	SegID determination depends on segmentation_update_map setting	
25:23	Sharpness Level		
	Format:	U3	
Specify the sharpness level, as one of regular deblocking strength control.			
Programming Notes			
Set to 0 to disable the use of sharpness control			
22:17	Filter Level		
	Format:	U6	
Specify the Filter level, as one of deblocking strength control			
Programming Notes			
Set to 0 to disable the use of level control			
16	Frame Parallel Decoding Mode		
Indicates if parallel decoding mode is enabled. This bit should come from Uncompressed header. Together with Error Resilient mode, they decide the value of AdaptProbabilityFlag.			
	Value	Name	
	0	Disable	

HCP_VP9_PIC_STATE									
	<table border="1"> <tr> <td style="width: 50px;">1</td> <td>Enable</td> </tr> </table>	1	Enable						
1	Enable								
15	<p>Error Resilient Mode Indicates if error resilient mode is enabled. This bit should come from Uncompressed header. When error resilient is 1, Frame Parallel Decoding Mode will be 1, and Refresh Frame Context will be 0. When error resilient is 0, Frame Parallel Decoding Mode and Refresh Frame Context read from bit stream. Together with Frame Parallel Decoding mode, they decide the value of AdaptProbabilityFlag.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Enable</td> </tr> </tbody> </table>	Value	Name	0	Disable	1	Enable		
Value	Name								
0	Disable								
1	Enable								
14	<p>Refresh Frame Context Indicates if Frame Context should be refresh. This bit should come from Uncompressed header</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Enable</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Decoder Only</p>	Value	Name	0	Disable	1	Enable		
Value	Name								
0	Disable								
1	Enable								
13	<p>Last Frame Type It indicates the frame type of previous frame (Key or Non-Key Frame)</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Key Frame</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Non Key Frame</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Not used in Encoder Mode</p>	Value	Name	0	Key Frame	1	Non Key Frame		
Value	Name								
0	Key Frame								
1	Non Key Frame								
12	<p>Selectable TX Mode</p> <table border="1"> <tr> <td style="width: 60px;">Format:</td> <td>U1</td> </tr> </table> <p>Indicates if tx_mode is selectable</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Encoder does not pack tu_size into bitstream. This helps reduce bitstream size further.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Encoder packs tu_size into bitstream.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>HW always picks tu_size from CU record of pak_obj. SW responsibility to set tu_size correct.</p>	Format:	U1	Value	Name	0	Encoder does not pack tu_size into bitstream. This helps reduce bitstream size further.	1	Encoder packs tu_size into bitstream.
Format:	U1								
Value	Name								
0	Encoder does not pack tu_size into bitstream. This helps reduce bitstream size further.								
1	Encoder packs tu_size into bitstream.								

HCP_VP9_PIC_STATE													
11	Hybrid Prediction Mode Format: U1 Indicates if comp_pred_mode is hybrid <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 90%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>comp_prediction_mode!=HYBRID, Encoder does not pack comp_pred_mode [interpred_comp in pak_obj] into bitstream.</td> </tr> <tr> <td>1</td> <td>comp_prediction_mode==HYBRID, Encoder packs comp_pred_mode into bitstream. This helps reduce bitstream size further.</td> </tr> </tbody> </table>	Value	Name	0	comp_prediction_mode!=HYBRID, Encoder does not pack comp_pred_mode [interpred_comp in pak_obj] into bitstream.	1	comp_prediction_mode==HYBRID, Encoder packs comp_pred_mode into bitstream. This helps reduce bitstream size further.						
	Value	Name											
	0	comp_prediction_mode!=HYBRID, Encoder does not pack comp_pred_mode [interpred_comp in pak_obj] into bitstream.											
	1	comp_prediction_mode==HYBRID, Encoder packs comp_pred_mode into bitstream. This helps reduce bitstream size further.											
	10	Use Prev in Find MV References Format: Enable 0: Temporal MV buffer is not available for MV prediction 1: Temporal MV buffer is available for MV prediction This is set to 0 when: The last picture has a different size Current picture is error-resilient mode Current picture is intra_only, or keyframe Last picture was intra_only or keyframe Last picture was not a displayed picture.											
Ref Frame Sign Bias[0..2] Format: U3 Reference Frame sign bias (not including intra reference) Bit[7] Sign Bias of Last Frame Bit[8] Sign Bias of Golden Frame Bit[9] Sign Bias of AltRef Frame													
6:4	Mcomp Filter Type Format: U3 Indicate Motion Compensation Filter type. If set to 4, encoder uses modes in pak_obj command.												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Eight-tap</td> </tr> <tr> <td>1h</td> <td>Eight-tap-Smooth</td> </tr> <tr> <td>2h</td> <td>Eight-tap-Sharp</td> </tr> <tr> <td>3h</td> <td>Bilinear</td> </tr> <tr> <td>4h</td> <td>Switchable</td> </tr> </tbody> </table>	Value	Name	0h	Eight-tap	1h	Eight-tap-Smooth	2h	Eight-tap-Sharp	3h	Bilinear	4h	Switchable
	Value	Name											
	0h	Eight-tap											
1h	Eight-tap-Smooth												
2h	Eight-tap-Sharp												
3h	Bilinear												
4h	Switchable												
3	Allow Hi Precision MV Format: Enable Indicate high precision mode for Motion Vector prediction												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Normal mode</td> </tr> <tr> <td>1h</td> <td>High Precision mode</td> </tr> </tbody> </table>	Value	Name	0h	Normal mode	1h	High Precision mode						
	Value	Name											
0h	Normal mode												
1h	High Precision mode												
2	IntraOnly Flag												

HCP_VP9_PIC_STATE																			
		<table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Indicates intra-only for inter pics. MBZ for keyframes.</p> <p style="text-align: center;">Programming Notes</p> <p>Used for Non-displayable picture; SW responsibility to make sure no Inter block in pak_obj of this frame</p> <table border="1"> <tr> <th>Value</th> <th>Description</th> </tr> <tr> <td>0</td> <td>Inter Frame use both intra/inter-blocks</td> </tr> <tr> <td>1</td> <td>Inter frame use only intra-blocks</td> </tr> </table>	Format:	Enable	Value	Description	0	Inter Frame use both intra/inter-blocks	1	Inter frame use only intra-blocks									
	Format:	Enable																	
	Value	Description																	
0	Inter Frame use both intra/inter-blocks																		
1	Inter frame use only intra-blocks																		
1	<p>Adapt Probabilities Flag</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Indicates that the probabilities used to decode this frame should be adapted</p> <table border="1"> <tr> <th>Value</th> <th>Name</th> </tr> <tr> <td>0h</td> <td>0: Do not adapt (error resilient or frame_parallel_mode are set)</td> </tr> <tr> <td>1h</td> <td>1: Adapt (not error resilient and not frame_parallel_mode)</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>Only forward adaptation is supported in encoder mode.</p>	Format:	Enable	Value	Name	0h	0: Do not adapt (error resilient or frame_parallel_mode are set)	1h	1: Adapt (not error resilient and not frame_parallel_mode)										
Format:	Enable																		
Value	Name																		
0h	0: Do not adapt (error resilient or frame_parallel_mode are set)																		
1h	1: Adapt (not error resilient and not frame_parallel_mode)																		
0	<p>Frame Type</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>Specifies the VP9 frame type</p> <table border="1"> <tr> <th>Value</th> <th>Name</th> </tr> <tr> <td>0h</td> <td>Key Frame</td> </tr> <tr> <td>1h</td> <td>Inter Frame</td> </tr> </table>	Format:	U1	Value	Name	0h	Key Frame	1h	Inter Frame										
Format:	U1																		
Value	Name																		
0h	Key Frame																		
1h	Inter Frame																		
3	31:28	<p>Profile Level</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> </table> <p>This indicates VP9 Profile level from bitstream</p> <table border="1"> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> <tr> <td>0</td> <td>Profile_0</td> <td>Profile 0 only supports 8 bit 420 only</td> </tr> <tr> <td>2</td> <td>Profile_2</td> <td>Profile 2 only supports 10 bits 420 only</td> </tr> <tr> <td>1</td> <td>Profile_1</td> <td>Profile 1 only supports 8 bit 444 only</td> </tr> <tr> <td>3</td> <td>Profile_3</td> <td>Profile 3 only supports 10-bit 444 only</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>Profile 0, 1, 2, and 3 are supported. Profile 0: 8 bit 420 only Profile 1: 8 bit 444 (422 is NOT supported) Profile 2: 10/12 bit 420 Profile 3: 10/12 bit 444 (422 is NOT supported)</p>	Format:	U4	Value	Name	Description	0	Profile_0	Profile 0 only supports 8 bit 420 only	2	Profile_2	Profile 2 only supports 10 bits 420 only	1	Profile_1	Profile 1 only supports 8 bit 444 only	3	Profile_3	Profile 3 only supports 10-bit 444 only
Format:	U4																		
Value	Name	Description																	
0	Profile_0	Profile 0 only supports 8 bit 420 only																	
2	Profile_2	Profile 2 only supports 10 bits 420 only																	
1	Profile_1	Profile 1 only supports 8 bit 444 only																	
3	Profile_3	Profile 3 only supports 10-bit 444 only																	

HCP_VP9_PIC_STATE													
27:24	BitDepthMinus8 Format: U4 This indicates the bitdepth (minus 8) of the pixels												
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Bitdepth_8</td> <td>It indicates pixel bitdepth is 8. Only profile 0 is allowed in this mode. It indicates pixel bitdepth is 8. Only profile 0 and 1 are allowed in this mode.</td> </tr> <tr> <td style="text-align: center;">2</td> <td>Bitdepth_10</td> <td>It indicates pixel bitdepth is 10. Only profile 2 is allowed in this mode. It indicates pixel bitdepth is 10. Only profile 2 and 3 are allowed in this mode.</td> </tr> <tr> <td style="text-align: center;">4</td> <td>Bitdepth_12</td> <td>It indicates pixel bitdepth is 12. Only profile 2 is allowed in this mode.</td> </tr> </tbody> </table>	Value	Name	Programming Notes	0	Bitdepth_8	It indicates pixel bitdepth is 8. Only profile 0 is allowed in this mode. It indicates pixel bitdepth is 8. Only profile 0 and 1 are allowed in this mode.	2	Bitdepth_10	It indicates pixel bitdepth is 10. Only profile 2 is allowed in this mode. It indicates pixel bitdepth is 10. Only profile 2 and 3 are allowed in this mode.	4	Bitdepth_12	It indicates pixel bitdepth is 12. Only profile 2 is allowed in this mode.
	Value	Name	Programming Notes										
	0	Bitdepth_8	It indicates pixel bitdepth is 8. Only profile 0 is allowed in this mode. It indicates pixel bitdepth is 8. Only profile 0 and 1 are allowed in this mode.										
2	Bitdepth_10	It indicates pixel bitdepth is 10. Only profile 2 is allowed in this mode. It indicates pixel bitdepth is 10. Only profile 2 and 3 are allowed in this mode.											
4	Bitdepth_12	It indicates pixel bitdepth is 12. Only profile 2 is allowed in this mode.											
Programming Notes													
In profile 0 and 1, only value of 0 (8 bit pixel) is allowed. In profile 2 and 3, only value of 2 and 4 (10 or 12 bit pixel) are allowed.													
23:22	Chroma Sampling Format Format: U2 This indicates the chroma sampling format of the bitstream												
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Format_420</td> <td>Chroma Format 420, supported by profile 0 and 2</td> </tr> <tr> <td style="text-align: center;">2</td> <td>Format_444</td> <td>Chroma Format 444, supported by Profile 1 and 3</td> </tr> </tbody> </table>	Value	Name	Programming Notes	0	Format_420	Chroma Format 420, supported by profile 0 and 2	2	Format_444	Chroma Format 444, supported by Profile 1 and 3			
	Value	Name	Programming Notes										
	0	Format_420	Chroma Format 420, supported by profile 0 and 2										
2	Format_444	Chroma Format 444, supported by Profile 1 and 3											
Programming Notes													
Currently only 420 and 444 are supported (in profile 0, 1, 2 and 3). All other modes are not valid. Value 0: Format 420: Profile 0 and 2 only (supported) Value 1: Format 422: Profile 1 and 3 only: Currently NOT supported Value 2: Format 444: Profile 1 and 3 only:(supported)													
21	SSE Enable Format: Enable This field indicates if SSE statistics generation is enabled in the PAK. If enabled, the classID in the CU packet is used along with the SSE thresholds in the picture state to generate the SSE frame statistics. When enabled, the PAK will also write and read source pixels (similar to the reference pixels in the LCU ILDB streamout) to generate the SSE distortion. Valid only in VDenc mode. Encoder Only												
20:10	Reserved												

HCP_VP9_PIC_STATE																						
		<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																		
Format:	MBZ																					
	9:8	<p>Log2 Tile Row</p> <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <p>This indicates the number of tile rows (log2).</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>1 Tile Row</td> </tr> <tr> <td>1h</td> <td>2 Tile Row</td> </tr> <tr> <td>2h</td> <td>4 Tile Row</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Decoder Only as encoder must use TILE_CODING_COMMAND</td> </tr> </tbody> </table>	Format:	U2	Value	Name	0h	1 Tile Row	1h	2 Tile Row	2h	4 Tile Row	Programming Notes	Decoder Only as encoder must use TILE_CODING_COMMAND								
Format:	U2																					
Value	Name																					
0h	1 Tile Row																					
1h	2 Tile Row																					
2h	4 Tile Row																					
Programming Notes																						
Decoder Only as encoder must use TILE_CODING_COMMAND																						
	7:4	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																		
Format:	MBZ																					
	3:0	<p>Log2 Tile Column</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> </table> <p>This indicates the number of tile rows (log2).</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>1 Tile Column</td> </tr> <tr> <td>1h</td> <td>2 Tile Column</td> </tr> <tr> <td>2h</td> <td>4 Tile Column</td> </tr> <tr> <td>3h</td> <td>8 Tile Column</td> </tr> <tr> <td>4h</td> <td>16 Tile Column</td> </tr> <tr> <td>5h</td> <td>32 Tile Column</td> </tr> <tr> <td>6h</td> <td>64 Tile Column</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Decoder only as encoder must use TILE_CODING_COMMAND</td> </tr> </tbody> </table>	Format:	U4	Value	Name	0h	1 Tile Column	1h	2 Tile Column	2h	4 Tile Column	3h	8 Tile Column	4h	16 Tile Column	5h	32 Tile Column	6h	64 Tile Column	Programming Notes	Decoder only as encoder must use TILE_CODING_COMMAND
Format:	U4																					
Value	Name																					
0h	1 Tile Column																					
1h	2 Tile Column																					
2h	4 Tile Column																					
3h	8 Tile Column																					
4h	16 Tile Column																					
5h	32 Tile Column																					
6h	64 Tile Column																					
Programming Notes																						
Decoder only as encoder must use TILE_CODING_COMMAND																						
4	31:16	<p>Horizontal Scale Factor for LAST</p> <table border="1"> <tr> <td>Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the last reference frame Set to $(LastWidth * 2^{14})/CurrentWidth$</p>	Format:	U2.14																		
	Format:	U2.14																				
15:0	<p>Vertical Scale Factor for LAST</p> <table border="1"> <tr> <td>Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the last reference frame Set to $(LastHeight * 2^{14})/CurrentHeight$</p>	Format:	U2.14																			
Format:	U2.14																					
5	31:16	<p>Horizontal Scale Factor for GOLDEN</p>																				

HCP_VP9_PIC_STATE							
		<table border="1"> <tr> <td>Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the golden reference frame Set to $(LastWidth * 2^{14})/CurrentWidth$</p>	Format:	U2.14			
Format:	U2.14						
	15:0	<p>Vertical Scale Factor for GOLDEN</p> <table border="1"> <tr> <td>Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the golden reference frame Set to $(LastHeight * 2^{14})/CurrentHeight$</p>	Format:	U2.14			
Format:	U2.14						
6	31:16	<p>Horizontal Scale Factor for ALTREF</p> <table border="1"> <tr> <td>Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the altref reference frame Set to $(LastWidth * 2^{14})/CurrentWidth$</p>	Format:	U2.14			
	Format:	U2.14					
15:0	<p>Vertical Scale Factor for ALTREF</p> <table border="1"> <tr> <td>Format:</td> <td>U2.14</td> </tr> </table> <p>This indicates the scaling factor between current frame and the altref reference frame Set to $(LastHeight * 2^{14})/CurrentHeight$</p>	Format:	U2.14				
Format:	U2.14						
7	31	Reserved					
	30	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
	Format:	MBZ					
	29:16	<p>Last Frame Height In Pixels Minus 1</p> <table border="1"> <tr> <td>Format:</td> <td>U14</td> </tr> </table> <p>Specifies the height of the Last picture in units of pixels. The Last picture height in units of luma samples equals $(LastFrameHeightInMinBlocksMinus1 + 1)$</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-16383</td> <td>1-16K Pixels (encoder only goes to 8K)</td> </tr> </tbody> </table>	Format:	U14	Value	Description	0-16383
Format:	U14						
Value	Description						
0-16383	1-16K Pixels (encoder only goes to 8K)						
15:14	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
13:0	<p>Last Frame Width In Pixels Minus 1</p> <table border="1"> <tr> <td>Format:</td> <td>U14</td> </tr> </table> <p>Specifies the width of the Last picture in units of pixels. The Last picture width in units of luma samples equals $(LastFrameWidthInMinBlocksMinus1 + 1)$</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-16383</td> <td>1-16K Pixels (encoder only goes to 8K)</td> </tr> </tbody> </table>	Format:	U14	Value	Description	0-16383	1-16K Pixels (encoder only goes to 8K)
Format:	U14						
Value	Description						
0-16383	1-16K Pixels (encoder only goes to 8K)						
8	31:30	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ						

HCP_VP9_PIC_STATE								
	29:16	Golden Frame Height In Pixels Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U14</td> </tr> </table> <p>Specifies the height of the Last picture in units of pixels. The Golden picture height in units of luma samples equals (LastFrameHeightInMinBlocksMinus1 + 1)</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-16383</td> <td>1-16K Pixels (encoder only goes to 8K)</td> </tr> </tbody> </table>	Format:	U14	Value	Description	0-16383	1-16K Pixels (encoder only goes to 8K)
	Format:	U14						
	Value	Description						
0-16383	1-16K Pixels (encoder only goes to 8K)							
15:14	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ					
Format:	MBZ							
13:0	Golden Frame Width In Pixels Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U14</td> </tr> </table> <p>Specifies the width of the Last picture in units of pixels. The Golden picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1 + 1)</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-16383</td> <td>1-16K Pixels (encoder only goes to 8K)</td> </tr> </tbody> </table>	Format:	U14	Value	Description	0-16383	1-16K Pixels (encoder only goes to 8K)	
Format:	U14							
Value	Description							
0-16383	1-16K Pixels (encoder only goes to 8K)							
9	31:30	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	29:16	Altref Frame Height In Pixels Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U14</td> </tr> </table> <p>Specifies the height of the Last picture in units of pixels. The Altref picture height in units of luma samples equals (LastFrameHeightInMinBlocksMinus1 + 1)</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-16383</td> <td>1-16K Pixels (encoder only goes to 8K)</td> </tr> </tbody> </table>	Format:	U14	Value	Description	0-16383	1-16K Pixels (encoder only goes to 8K)
	Format:	U14						
Value	Description							
0-16383	1-16K Pixels (encoder only goes to 8K)							
15:14	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ					
Format:	MBZ							
13:0	Altref Frame Width In Pixels Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U14</td> </tr> </table> <p>Specifies the width of the Last picture in units of pixels. The Altref picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1 + 1)</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-16383</td> <td>1-16K Pixels (encoder only goes to 8K)</td> </tr> </tbody> </table>	Format:	U14	Value	Description	0-16383	1-16K Pixels (encoder only goes to 8K)	
Format:	U14							
Value	Description							
0-16383	1-16K Pixels (encoder only goes to 8K)							
10	31:16	First Partition Size in Bytes [15:0] <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U16</td> </tr> </table> <p>Specifies the number of bytes taken up by the first partition size which handle the</p>	Format:	U16				
Format:	U16							

HCP_VP9_PIC_STATE										
		probability updates <div style="border: 1px solid black; background-color: #e6f2ff; padding: 2px; text-align: center;">Programming Notes</div> Only used by Decoder								
	15:8	Reserved Format: MBZ								
	7:0	Uncompressed Header Length in Bytes [7:0] Format: U8 Specifies the number of bytes taken up by the uncompressed frame header. <div style="border: 1px solid black; background-color: #e6f2ff; padding: 2px; text-align: center;">Programming Notes</div> this field is used by decoder only								
11	31:4	Reserved Format: MBZ								
	3	Reserved								
	2	Reserved								
	1	Motion Comp Scaling Enable Bit This bit must be set to "1" <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Enable [Default]</td> <td>This enables Motion Comp Scaling</td> </tr> </tbody> </table>	Value	Name	Description	1	Enable [Default]	This enables Motion Comp Scaling		
	Value	Name	Description							
1	Enable [Default]	This enables Motion Comp Scaling								
0	Reserved Format: MBZ									
12	31:0	Reserved Format: MBZ								
13	31:26	Reserved Format: MBZ								
	25	Header Insertion Enable Format: U1 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 10%;">Name</th> <th style="width: 80%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td></td> <td>No header insertion into the output bitstream buffer, before the current slice encoded bits.</td> </tr> <tr> <td style="text-align: center;">1</td> <td></td> <td>Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits.</td> </tr> </tbody> </table> <p>Must be followed by the PAK Insertion Object Command to perform the actual insertion. For VP9: Header is always present and this bit can never be Zero. As HW does the header back-annotation at the end of frame we currently cannot disable it, if header was not written by HW. Media SDK sends 2 headers. One header has original header and second header has 0s for BRC parameters (LF refDelta, ModeDelta and BaseQindex). Driver needs</p>	Value	Name	Description	0		No header insertion into the output bitstream buffer, before the current slice encoded bits.	1	
Value	Name	Description								
0		No header insertion into the output bitstream buffer, before the current slice encoded bits.								
1		Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits.								

HCP_VP9_PIC_STATE																
	<p>to pick first header for the first pass, and second header for subsequent passes.</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Programming Notes		Encoder Only												
Programming Notes																
Encoder Only																
	<p>24 Tail Insertion Enable</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: right;">U1</td> </tr> </table> <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>No tail insertion into the output bitstream buffer, after the current slice encoded bits.</td> </tr> <tr> <td>1</td> <td></td> <td>Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.</td> </tr> </tbody> </table> <p>Must be followed by the PAK Insertion Object Command to perform the actual insertion. Tail has to be inserted only with the last slice of frame and for VP9 only at the end of Frame.</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U1	Value	Name	Description	0		No tail insertion into the output bitstream buffer, after the current slice encoded bits.	1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.	Programming Notes		Encoder Only	
Format:	U1															
Value	Name	Description														
0		No tail insertion into the output bitstream buffer, after the current slice encoded bits.														
1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.														
Programming Notes																
Encoder Only																
	<p>23:16 Base Q Index (Same as Luma AC)</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: right;">U8</td> </tr> </table> <p>Added to Delta Q index of Segment Valid Values : 0..255</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U8	Programming Notes		Encoder Only										
Format:	U8															
Programming Notes																
Encoder Only																
	<p>15:0 Compressed header BIN count</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: right;">U16</td> </tr> </table> <p>Number of bins compressed header This field is fixed inside HW</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U16	Programming Notes		Encoder Only										
Format:	U16															
Programming Notes																
Encoder Only																
14	<p>31:21 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: right;">MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ														
	<p>20:16 Luma DC Q Index Delta</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: right;">S4</td> </tr> </table> <p>QP delta value for Luma DC Valid Values : -15..15</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	S4	Programming Notes		Encoder Only										
Format:	S4															
Programming Notes																
Encoder Only																
<p>15:13 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: right;">MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ															

HCP_VP9_PIC_STATE	
	12:8 ChromaDC_QindexDelta
	Format: S4
	QP Delta Value For Chroma DC Valid Values : -15..15
	Programming Notes
	Encoder Only
	7:5 Reserved
	Format: MBZ
	4:0 ChromaAC_QindexDelta
	Format: S4
	QP Delta Value For Chroma AC Valid Values : -15..15
	Programming Notes
	Encoder Only
15	31 Reserved
	Format: MBZ
	30:24 LF_ref_delta3
	Format: S6
	Loop filter level delta3 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31.
	Programming Notes
	Encoder Only
	23 Reserved
Format: MBZ	
	22:16 LF_ref_delta2
	Format: S6
	Loop filter level delta2 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31.
	Programming Notes
	Encoder Only
	15 Reserved
	Format: MBZ
	14:8 LF_ref_delta1
	Format: S6
	Loop filter level delta1 value; valid range -6363

HCP_VP9_PIC_STATE								
		<p>With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31.</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Programming Notes		Encoder Only			
Programming Notes								
Encoder Only								
	7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
	6:0	<p>LF_ref_delta0</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>S6</td> </tr> </table> <p>Loop filter level delta0 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31.</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	S6	Programming Notes		Encoder Only	
Format:	S6							
Programming Notes								
Encoder Only								
16	31:15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
	14:8	<p>LF Mode Delta 1</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>S6</td> </tr> </table> <p>Loop filter level mode1 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31.</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	S6	Programming Notes		Encoder Only	
Format:	S6							
Programming Notes								
Encoder Only								
	7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
	6:0	<p>LF Mode Delta 0</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>S6</td> </tr> </table> <p>Loop filter level mode1 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31.</p> <table border="1" style="width: 100%;"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	S6	Programming Notes		Encoder Only	
Format:	S6							
Programming Notes								
Encoder Only								
17	31:16	<p>BitOffsetForLFModeDelta</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U16</td> </tr> </table> <p>Offset from starting position of output bitstream in bits where LFModeDelta should be inserted. In BRC mode, always insert LFModeDelta. (This implies uncompressed header</p>	Format:	U16				
Format:	U16							

HCP_VP9_PIC_STATE												
		<p>should have: mode_ref_delta_enabled=1 and mode_ref_delta_update=1} and The uncompressed header starting from this offset BitOffsetForLFModeDelta has to have the following 16 bits format: {Start here: 1b1, mode_delta_0[6:0], 1b1, mode_delta_1[6:0]} and BitOffsetForLFModeDelta = BitOffsetForLFRefDelta + 32. Encoder Only</p>										
	15:0	<p>BitOffsetForLFRefDelta</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Offset from starting position of output bitstream in bits where LFRefDelta should be inserted. In BRC mode, always insert LFRefDelta. {This implies uncompressed header should have: mode_ref_delta_enabled=1 and mode_ref_delta_update=1} and The uncompressed header starting from this offset BitOffsetForLFRefDelta has to have the following 32 bits format: {Start here: 1b1, ref_delta_0[6:0], 1b1, ref_delta_1[6:0], 1b1, ref_delta_2[6:0], 1b1, ref_delta_3[6:0]}. Encoder Only</p>	Format:	U16								
Format:	U16											
18	31:16	<p>BitOffsetForLFLevel</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Offset from starting position of output bitstream in bits where LFLevel should be inserted. Encoder Only</p>	Format:	U16								
	Format:	U16										
15:0	<p>BitOffsetForQindex</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table> <p>Offset from starting position of output bitstream in bits where Qindex should be inserted. Encoder Only</p>	Format:	U16									
Format:	U16											
19	31:27	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ								
	Format:	MBZ										
26	<p>FrameSzUnderStatusEn - FrameBitRateMinReportMask</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> <td>Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_VP9_IMAGE_STATUS control register.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Enable</td> <td>Set bit 2 (Frame Bit Count Violate -- under run) of</td> </tr> </tbody> </table>	Format:	U1	Value	Name	Description	0	Disable	Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_VP9_IMAGE_STATUS control register.	1	Enable	Set bit 2 (Frame Bit Count Violate -- under run) of
Format:	U1											
Value	Name	Description										
0	Disable	Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_VP9_IMAGE_STATUS control register.										
1	Enable	Set bit 2 (Frame Bit Count Violate -- under run) of										

HCP_VP9_PIC_STATE		
		HCP_VP9_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit Rate Minimum limit.
	Programming Notes	
	Encoder Only	
25	FrameSzOverStatusEn - FrameBitRateMaxReportMask	
	Format:	U1
	This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax.	
	Value	Name Description
	0	Disable Do not update bit 1 of HCP_VP9_IMAGE_STATUS control register.
	1	Enable Set bit 1 of HCP_VP9_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit Rate Maximum limit.
	Programming Notes	
	Encoder Only	
24:18	Reserved	
	Format:	MBZ
17	Reserved	
	Format:	MBZ
16	NonFirstPassFlag	
	This signals the current pass is not the first pass. It will imply designate HW behavior.	
	Value	Name Description
	0	Disable If it is initial-Pass, this bit is set to 0.
	1	Enable For subsequent passes, this bit is set to 1.
	Programming Notes	
	Encoder Only	
15:0	Reserved	
	Format:	MBZ
20	31	FrameBitrateMaxUnit
	Format:	U1
	This field is the Frame Bitrate Maximum Limit Units.	
	Value	Name Description
	0	Byte 32byte unit
	1	KiloByte 4Kbyte unit

HCP_VP9_PIC_STATE																	
		<table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Programming Notes		Encoder Only												
Programming Notes																	
Encoder Only																	
	30:14	Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	13:0	FrameBitRateMax <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">U14</td> </tr> </table> <p>This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value.</p> <p>0-512KB The programmable range is 0-512KB when FrameBitrateMaxUnit is 0.</p> <p>0-64MB The programmable range is 0-64Mbyte when FrameBitrateMaxUnit is 1.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U14	Programming Notes		Encoder Only										
Format:	U14																
Programming Notes																	
Encoder Only																	
21	31	FrameBitrateMinUnit <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">U1</td> </tr> </table> <p>This field is the Frame Bitrate Maximum Limit Units.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Byte</td> <td>32byte unit</td> </tr> <tr> <td>1</td> <td>KiloByte</td> <td>4Kbyte unit</td> </tr> </tbody> </table> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U1	Value	Name	Description	0	Byte	32byte unit	1	KiloByte	4Kbyte unit	Programming Notes		Encoder Only	
Format:	U1																
Value	Name	Description															
0	Byte	32byte unit															
1	KiloByte	4Kbyte unit															
Programming Notes																	
Encoder Only																	
	30:14	Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	13:0	FrameBitRateMin <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td style="text-align: center;">U14</td> </tr> </table> <p>This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value.</p> <p>0-512KB The programmable range is 0-512KB when FrameBitrateMinUnit is 0.</p> <p>0-64MB The programmable range is 0-64Mbyte when FrameBitrateMinUnit is 1.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U14	Programming Notes		Encoder Only										
Format:	U14																
Programming Notes																	
Encoder Only																	

HCP_VP9_PIC_STATE							
22..23	63:0	<p>FrameDeltaQindexMax</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>FrameDeltaQindexLFRange</td> </tr> </table> <p>Frame level delta Qindex which should be used in case FrameSize - FrameBitRateMax in the range of $((\text{FrameDeltaQindexLFMaxRange}[n] * \text{FrameBitRateMax} \gg 5))$, $\text{FrameDeltaQindexLFMaxRange}[n+1] * \text{FrameBitRateMax} \gg 5$). Each DelatQindexMax value is 8-bit with S7 format</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>If n == 7, DeltaQpMaxRange is infinity.</td> </tr> <tr> <td>Encoder Only</td> </tr> </table>	Format:	FrameDeltaQindexLFRange	Programming Notes	If n == 7, DeltaQpMaxRange is infinity.	Encoder Only
Format:	FrameDeltaQindexLFRange						
Programming Notes							
If n == 7, DeltaQpMaxRange is infinity.							
Encoder Only							
24	31:0	<p>FrameDeltaQindexMin</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>FrameDeltaQindexLFRange</td> </tr> </table> <p>Frame level delta Qindex which should be used in case FrameSize - FrameBitRateMin in the range of $((\text{FrameDeltaQindexLFMinRange}[n] * \text{FrameBitRateMin} \gg 5))$, $\text{FrameDeltaQindexLFMinRange}[n+1] * \text{FrameBitRateMin} \gg 5$). Each DelatQindexMin value is 8-bit with S7 format</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported)</td> </tr> <tr> <td>Encoder Only</td> </tr> </table>	Format:	FrameDeltaQindexLFRange	Programming Notes	If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported)	Encoder Only
Format:	FrameDeltaQindexLFRange						
Programming Notes							
If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported)							
Encoder Only							
25..26	63:0	<p>FrameDeltaLFMax</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>FrameDeltaQindexLFRange</td> </tr> </table> <p>Frame level delta Loop Filter Level which should be used in case FrameSize - FrameBitRateMax in the range of $((\text{FrameDeltaQindexLFMaxRange}[n] * \text{FrameBitRateMax} \gg 5))$, $\text{FrameDeltaQindexLFMaxRange}[n+1] * \text{FrameBitRateMax} \gg 5$). Each delta_if_max is 7 bits with S6 format [bits 7, 15, 23, 31,.63 are reserved]</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>If n == 7, FrameDeltaQindexLFMaxRange is infinity.</td> </tr> <tr> <td>Encoder Only</td> </tr> </table>	Format:	FrameDeltaQindexLFRange	Programming Notes	If n == 7, FrameDeltaQindexLFMaxRange is infinity.	Encoder Only
Format:	FrameDeltaQindexLFRange						
Programming Notes							
If n == 7, FrameDeltaQindexLFMaxRange is infinity.							
Encoder Only							
27	31:0	<p>FrameDeltaLFMin</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>FrameDeltaQindexLFRange</td> </tr> </table> <p>Frame level delta Loop Filter Level which should be used in case FrameSize - FrameBitRateMin in the range of $((\text{FrameDeltaQindexLFMinRange}[n] * \text{FrameBitRateMin} \gg 5))$, $\text{FrameDeltaQindexLFMinRange}[n+1] * \text{FrameBitRateMin} \gg 5$). Each delta_if_min is 7 bits with S6 format [bits 7, 15, 23, 31 are reserved]</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported)</td> </tr> <tr> <td>Encoder Only</td> </tr> </table>	Format:	FrameDeltaQindexLFRange	Programming Notes	If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported)	Encoder Only
Format:	FrameDeltaQindexLFRange						
Programming Notes							
If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported)							
Encoder Only							
28..29	63:0	<p>FrameDeltaQindexLFMaxRange</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>FrameDeltaQindexLFRange</td> </tr> </table>	Format:	FrameDeltaQindexLFRange			
Format:	FrameDeltaQindexLFRange						

HCP_VP9_PIC_STATE																							
		<p>Condition: FrameDeltaQindexLFMaxRange[n] >= FrameDeltaQindexLFMaxRange[n-1] This field is to calculate ranges for Frame level delta Qindex, specifically Frame level delta Qindex[n] and Frame level delta Qindex[n+1].</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">If n == 0, FrameDeltaQindexLFMaxRange is zero.</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Programming Notes		If n == 0, FrameDeltaQindexLFMaxRange is zero.		Encoder Only																
Programming Notes																							
If n == 0, FrameDeltaQindexLFMaxRange is zero.																							
Encoder Only																							
30	31:0	<p>FrameDeltaQindexLFMinRange</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>FrameDeltaQindexLFRange</td> </tr> </table> <p>Condition: FrameDeltaQindexLFMinRange[n] >= FrameDeltaQindexLFMinRange[n-1] This field is to calculate ranges for Frame level delta Qindex, specifically Frame level delta Qindex[n] and Frame level delta Qindex[n+1].</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">If n == 0, FrameDeltaQindexLFMinRange is zero.</td> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	FrameDeltaQindexLFRange	Programming Notes		If n == 0, FrameDeltaQindexLFMinRange is zero.		Encoder Only														
Format:	FrameDeltaQindexLFRange																						
Programming Notes																							
If n == 0, FrameDeltaQindexLFMinRange is zero.																							
Encoder Only																							
31	31:30	<p>MinFrameSizeUnits</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U2</td> </tr> </table> <p>This field is the Minimum Frame Size Units</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>4Kb</td> <td>Minimum Frame Size is in 4Kbytes.</td> </tr> <tr> <td>1</td> <td>16Kb</td> <td>Minimum Frame Size is in 4Kbytes.</td> </tr> <tr> <td>2</td> <td>Compatibility Mode</td> <td></td> </tr> <tr> <td>3</td> <td>6 Bytes</td> <td></td> </tr> </tbody> </table> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Encoder Only</td> </tr> </table>	Format:	U2	Value	Name	Description	0	4Kb	Minimum Frame Size is in 4Kbytes.	1	16Kb	Minimum Frame Size is in 4Kbytes.	2	Compatibility Mode		3	6 Bytes		Programming Notes		Encoder Only	
Format:	U2																						
Value	Name	Description																					
0	4Kb	Minimum Frame Size is in 4Kbytes.																					
1	16Kb	Minimum Frame Size is in 4Kbytes.																					
2	Compatibility Mode																						
3	6 Bytes																						
Programming Notes																							
Encoder Only																							
	29:16	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																			
Format:	MBZ																						
	15:0	<p>MinFramSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U16</td> </tr> </table> <p>Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done at the last slice of a picture. It is needed for CBR. Intel encoder parameter, not part of DXVA. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax. This field is reserved in Decode mode.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit)</td> </tr> </table>	Format:	U16	Programming Notes		Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit)																
Format:	U16																						
Programming Notes																							
Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit)																							

HCP_VP9_PIC_STATE		
		Programmable range is $0..(2^{16}-1) * 2^{14}$ when MinFrameSizeUnits is 1. (16KB unit) Encoder Only
32	31:16	Reserved Format: MBZ
	15:0	BitOffsetForFirstPartitionSize Format: U16 Offset from starting position of output bitstream in bits where First Partition Size should be inserted. Programming Notes Encoder Only
33	31:16	Class0_SSE_Threshold1 Format: U16 Programming Notes This field specifies the upper bound threshold for Class0 Zone1 to classify the per-4x4sblk SSE statistics. Class0_SSE_Threshold_0 < per-4x4sblk SSE <= Class0_SSE_Threshold_1 fall under Class0 Zone1. Class0_SSE_Threshold_1 < per-4x4sblk SSE fall under Class0 Zone2. Encoder Only
	15:0	Class0_SSE_Threshold0 Format: U16 Programming Notes This field specifies the upper bound threshold for Class0 Zone0 to classify the per-4x4sblk SSE statistics. per-4x4sblk SSE <= Class0_SSE_Threshold_0 fall under Class0 Zone0. Encoder Only
34..41 Programming Notes: SSE thresholds for Class 1-8, see DW 33 (SSE Class 0 thresholds) for format. Encoder Only	255:0	SSE thresholds for Class1-8 Format: U256

HCP_VP9_SEGMENT_STATE

HCP_VP9_SEGMENT_STATE			
Source:		VideoCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
22:16	Media Instruction Command		
	Default Value:	32h HCP_VP9_SEGMENT_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
1	31:3	Reserved	
		Format:	MBZ
	2:0	Segment ID	
Format:	U3		
The segment ID of the DWORDS following this one			
2	31:4	Reserved	
		Format:	MBZ
	3	Segment Reference Enabled	
Format:	Enable		
For encoder: When Segment Reference Enabled is set to 1, Software (Kernel) needs to program			

HCP_VP9_SEGMENT_STATE					
	<p>all PUs of this segment accordingly. This means: CU record PU refframe0=Segment Reference bit[2:1], refframe1=0, and interpret_comp=single.</p>				
2:1	<p>Segment Reference</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>Indicates reference frame for this segment.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table> <p>If the current frame is a KEY frame or INTRA_ONLY frame, this field should be set to INTRA for all segments.</p>	Format:	U2	Programming Notes	
Format:	U2				
Programming Notes					
0	<p>Segment Skipped</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Enable</td> </tr> </table> <p>Indicates skip mode for this segment.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table> <p>If set to 1, all delta coefficients and MVs within CU of this segment should be forced to zero in HW. No block less than 8x8 size allowed segment Skipped If set to 0, skipcoeff_flag should be coded as normal</p>	Format:	Enable	Programming Notes	
Format:	Enable				
Programming Notes					
3	<p>31:30 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ			
	<p>29:24 FilterLevelRef1Mode1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Decoder mode only</td> </tr> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>Indicates final filter level for Ref1 (Last Frame) and Mode 1.</p>	Exists If:	//Decoder mode only	Format:	U6
	Exists If:	//Decoder mode only			
	Format:	U6			
	<p>23:22 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ			
<p>21:16 FilterLevelRef1Mode0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Decoder mode only</td> </tr> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>Indicates final filter level for Ref1 (Last Frame) and Mode 0.</p>	Exists If:	//Decoder mode only	Format:	U6	
Exists If:	//Decoder mode only				
Format:	U6				
<p>15:14 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ				
<p>13:8 FilterLevelRef0Mode1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Decoder mode only</td> </tr> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>Indicates final filter level for Ref0 (Last Frame) and Mode 1.</p>	Exists If:	//Decoder mode only	Format:	U6	
Exists If:	//Decoder mode only				
Format:	U6				
<p>7:6 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ				

HCP_VP9_SEGMENT_STATE

	5:0	FilterLevelRef0Mode0	Exists If: //Decoder mode only	
			Format: U6	
		Indicates final filter level for Ref0 (Last Frame) and Mode 0.		
4	31:30	Reserved	Format:	MBZ
	29:24	FilterLevelRef3Mode1	Exists If: //Decoder mode only	
			Format: U6	
		Indicates final filter level for Ref3 (Last Frame) and Mode 1.		
	23:22	Reserved	Format:	MBZ
	21:16	FilterLevelRef3Mode0	Exists If: //Decoder mode only	
			Format: U6	
		Indicates final filter level for Ref3 (Last Frame) and Mode 0.		
	15:14	Reserved	Format:	MBZ
	13:8	FilterLevelRef2Mode1	Exists If: //Decoder mode only	
		Format: U6		
	Indicates final filter level for Ref2 (Last Frame) and Mode 1.			
	7:6	Reserved	Format:	MBZ
	5:0	FilterLevelRef2Mode0	Exists If: //Decoder mode only	
			Format: U6	
	Indicates final filter level for Ref2 (Last Frame) and Mode 0.			
5	31	Reserved	Format:	MBZ
	30:16	Luma AC Quant Scale (Decode mode Only)	Format:	U15
	Indicates final value of Luma AC Quantized Scale value.			

HCP_VP9_SEGMENT_STATE						
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[4,29247]</td> <td>Valid_Range</td> </tr> </tbody> </table>	Value	Name	[4,29247]	Valid_Range
Value	Name					
[4,29247]	Valid_Range					
	15	Reserved Format: _____ MBZ				
	14:0	Luma DC Quant Scale (Decode mode Only) Format: _____ U15 Indicates final value of Luma DC Quantized Scale value. <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[4,21387]</td> <td>Valid_Range</td> </tr> </tbody> </table>	Value	Name	[4,21387]	Valid_Range
Value	Name					
[4,21387]	Valid_Range					
6	31	Reserved Format: _____ MBZ				
	30:16	Chroma AC Quant Scale (Decode mode Only) Format: _____ U15 Indicates final value of Chroma AC Quantized Scale value. <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[4,29247]</td> <td>Valid_Range</td> </tr> </tbody> </table>	Value	Name	[4,29247]	Valid_Range
	Value	Name				
	[4,29247]	Valid_Range				
	15	Reserved Format: _____ MBZ				
	14:0	Chroma DC Quant Scale (Decode mode Only) Format: _____ U15 Indicates final value of Chroma DC Quantized Scale value. <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[4,21387]</td> <td>Valid_Range</td> </tr> </tbody> </table>	Value	Name	[4,21387]	Valid_Range
Value	Name					
[4,21387]	Valid_Range					
7	31:23	Reserved Format: _____ MBZ				
	22:16	Segment LF Level Delta (Encode mode Only) Format: _____ S6 Indicates the Loop Filter Delta for this segment. Must be 0 when segmentation_enabled == 0. Range -63..63				
	15:9	Reserved Format: _____ MBZ				
	8:0	Segment QIndex Delta (encode mode only) Format: _____ S8 Indicates the QIndex delta for this segment. Must be 0 when segmentation_enabled == 0. Range -255..255				

HCP_WEIGHTOFFSET_STATE

HCP_WEIGHTOFFSET_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>The HCP is selected with the Media Instruction Opcode "7h" for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.</p> <p>This slice level command is issued in both the encoding and decoding processes, if the weighted_pred_flag or weighted_bipred_flag equals one. If zero, then this command is not issued.</p> <p>Weight Prediction Values are provided in this command. Only Explicit Weight Prediction is supported in encoder.</p> <p>For P-Slice, this command is issued only once together with HCP_REF_IDX_STATE Command for L0 list. For B-Slice, this command can be issued up to two times together with HCP_REF_IDX_STATE Command, one for L0 list and one for L1 list.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name
		Format:	OpCode
		Codec/Engine Name = HCP = 7h	
22:16	Media Instruction Command		
	Default Value:	13h HCP_WEIGHTOFFSET_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
	28h	40	
1	31:1	Reserved	
		Format:	MBZ

HCP_WEIGHTOFFSET_STATE								
	0	RefPicListNum Format: U1 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%; text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Reference Picture List 0</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Reference Picture List 1</td> </tr> </tbody> </table>	Value	Name	0	Reference Picture List 0	1	Reference Picture List 1
Value	Name							
0	Reference Picture List 0							
1	Reference Picture List 1							
2..17	511:0	LumaOffsets Format: HCP_WEIGHTOFFSET_LUMA_ENTRY[16]						
18..33	511:0	ChromaOffsets Format: HCP_WEIGHTOFFSET_CHROMA_ENTRY[16]						
34..41	255:0	ChromaOffsetsExt Format: HCP_WEIGHTOFFSET_CHROMA_EXT_ENTRY[8]						

HEVC_SFC_AVS_CHROMA_Coeff_Table

HEVC_SFC_AVS_CHROMA_Coeff_Table			
Source:	BSpec		
Length Bias:	2		
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
		26:23	Media Command Opcode
	Default Value:		9h Media HEVC+SFC Mode
		Format:	OpCode
		22:21	SubOpcodeA
	Default Value:		0h Common
		Format:	OpCode
		20:16	SubOpcodeB
	Default Value:		6h SFC_AVS CHROMA Coeff_Table
	Format:	OpCode	
	15:12	Reserved	
Format:		MBZ	
11:0	DWord Length		
	Default Value:	3Fh Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..64	2047:0	AVS CHROMA Coefficient Table Body	
		Format:	SFC_AVS_CHROMA_COEFF_TABLE_BODY

HEVC_SFC_AVS_LUMA_Coeff_Table

HEVC_SFC_AVS_LUMA_Coeff_Table			
Source:	BSpec		
Length Bias:	2		
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
	26:23	Media Command Opcode	
		Default Value:	9h Media HEVC+SFC Mode
	22:21	SubOpcodeA	
		Default Value:	0h Common
	20:16	SubOpcodeB	
		Default Value:	5h SFC_AVS LUMA Coeff_Table
	15:12	Reserved	
		Format:	MBZ
	11:0	DWord Length	
Default Value:		7Fh Excludes DWord (0,1)	
Format:		=n	
Total Length - 2			
1..128	4095:0	AVS LUMA Coefficient Table Body	
		Format:	SFC_AVS_LUMA_COEFF_TABLE_BODY

HEVC_SFC_AVS_STATE

HEVC_SFC_AVS_STATE			
Source:	BSpec		
Length Bias:	2		
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:23	Media Command Opcode	
		Format:	OpCode
		Value	Name
		9h	Media HEVC+SFC Mode [Default]
22:21	SubOpcodeA		
	Default Value:	0h Common	
	Format:	OpCode	
20:16	SubOpcodeB		
	Default Value:	2h SFC_AVS_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	2h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1..3	95:0	AVS State Body	
	Format:	SFC_AVS_STATE_BODY	

HEVC_SFC_FRAME_START

HEVC_SFC_FRAME_START			
Source:	BSpec		
Length Bias:	2		
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:23	Media Command Opcode	
		Default Value:	9h Media HEVC+SFC Mode
		Format:	OpCode
	22:21	SubOpcodeA	
Default Value:		0h Common	
Format:		OpCode	
20:16	SubOpcodeB		
	Default Value:	4h SFC_FRAME_START	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		
1	31:0	Frame Start Body	
		Format:	SFC_FRAME_START_BODY

HEVC_SFC_IEF_STATE

HEVC_SFC_IEF_STATE			
Source:	BSpec		
Length Bias:	2		
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:23	Media Command Opcode	
		Default Value:	9h Media HEVC+SFC Mode
		Format:	OpCode
	22:21	SubOpcodeA	
Default Value:		0h Common	
Format:		OpCode	
20:16	SubOpcodeB		
	Default Value:	3h SFC_IEF_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	11:0	DWord Length	
		Default Value:	16h Excludes DWord (0,1)
		Format:	=n
		Total Length - 2	
1..23	735:0	SFC IEF State Body	
		Format:	SFC_IEF_STATE_BODY

HEVC_VP9_RDOQ_STATE

HEVC_VP9_RDOQ_STATE				
Source:		VideoCS		
Length Bias:		2		
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h	
		Format:	Opcode	
	PARALLEL_VIDEO_PIPE			
	28:27	Pipeline		
		Default Value:	2h	
		Format:	Opcode	
	MFX_COMMON			
26:23	Opcode			
	Default Value:	7h		
	Format:	Opcode		
Codec/Engine Name = HCP = 7h				
22:21	SubOpA			
	Default Value:	0h		
Format:		Opcode		
20:16	SubOpB			
	Default Value:	8h		
Format:		Opcode		
15:12	Reserved			
	Format:	MBZ		
11:0	DWord Length			
	Total Length - 2			
	Value	Name	Description	
98h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)		
1	31	Disable HTQ performance fix0 set to disable performance optimizations done by doubling LNZ and OSR storage Set to 1, to go back to single LNZ and OSR (no optimization) Set to 0, to use double buffer LNZ and OSR		
	30	Disable HTQ performance fix1 set to disable performance optimization done to save 1clk while switching from EBB to GTRAM		

HEVC_VP9_RDOQ_STATE				
		<p>storage. This is critical for IntraLoop performance Set to 1, disable optimization Set to 0, enable optimization</p>		
	29:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
2..33	1023:0	<p>IntraLumaLambda_QP0_63</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]			
34..65	1023:0	<p>IntraChromaLambda_QP0_63</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]			
66..97	1023:0	<p>InterLumaLambda_QP0_63</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]			
98..129	1023:0	<p>InterChromaLambda_QP0_63</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]			
130..135	191:0	<p>IntraLumaLambda_QP64_75</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]			
136..141	191:0	<p>IntraChromaLambda_QP64_75</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]			
142..147	191:0	<p>InterLumaLambda_QP64_75</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]			
148..153	191:0	<p>InterChromaLambda_QP64_75</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]</td> </tr> </table>	Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]
Format:	HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]			

HI8DS Render Target Write MSD

MSD_RTW_HI8DS - HI8DS Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 0h Format: Opcode Full precision data message
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. Programming Notes This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	Message Type Default Value: 0Ch

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

		Format:	Opcode
		Render Target Write message	
13	Per-Sample PS Enable		
		Format:	Enable
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	<p>This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.</p> <p>When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.</p> <p>When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).</p> <p>When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.</p>		
12	Last Render Target Select		
		Format:	Enable
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select		
		Format:	MDC_RT_SGS
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype		
		Default Value:	3h
		Format:	Opcode
	SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].		
7:0	Binding Table Index		
		Format:	MDC_BTS
	Specifies the Binding Table Index for the message		

If

if - If			
Source:	Eulsa		
Length Bias:	4		
<p>An if instruction starts an if/endif or an if/else/endif block of code. It restricts execution within the conditional block to only those channels that were enabled via the predicate control. Each if instruction must have a matching endif instruction and may have up to one matching else instruction before the matching endif. If all channels are inactive (for the if/endif or if/else/endif block), a jump is performed to the instruction referenced by JIP. This jump must be to right after the matching else instruction when present, or otherwise to the matching endif instruction of the conditional block. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.</p>			
<p>The following table describes the 32-bit exit code <JIP> and <UIP>. If <branch_ctrl> is set, then the JIP points to the first join instruction within the if block. If <branch_ctrl> is not set, <JIP> should point to the instruction right after the matching else instruction if it exists, otherwise <JIP> should point to the endif instruction. <UIP> should always point to the endif instruction. When a jump occurs, this value is added to IP pre-increment. In instruction binary, <JIP> and <UIP> are at location <src0> & <src1> and must be of type D (signed dword integer).</p>			
<p>Format:</p> <pre>[(pred)] if (exec_size JIP UIP <branch_ctrl></pre>			
Restriction			
The execution size must be the same for the if, else, and endif instructions of the same code block.			
Syntax			
[(pred)] if (exec_size) imm32 imm32 <branch_ctrl>			
Pseudocode			
<pre>Evaluate(WrEn);for (n = 0; n < 32; n++) { if (WrEn.channel[n] == 0) { PcIP[n] = IP + JIP; } else { PcIP[n] = IP + 1; }}if (PcIP != (IP + 1)) { // for all channels Jump(IP + JIP);}</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
	Format:	MBZ	
	127:96	JIP	
Exists If:		([Src0.IsImm]==true)	
Format:	S31		

if - If

		The byte-aligned jump distance if a jump is taken for the channel.	
95:80	Reserved		
	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
95:64	Reserved		
	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)	
	Format:	MBZ	
95:64	UIP		
	Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)	
	Format:	S31	
	The byte aligned jump distance if a jump is taken for the instruction.		
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false)	
	Format:	DirectOperand	
65:64	Reserved		
	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
63:50	Dst.Operand		
	Format:	DirectOperand	
49:48	Reserved		
	Format:	MBZ	
47	Src1.IsImm	This field indicate that Source 1 operand is carrying an immediate value	
		Value	Name
		0	false
		1	true
46	Src0.IsImm	This field indicate that Source 0 operand is carrying an immediate value	
		Value	Name
		0	false
		1	true
45:34	Reserved		
	Format:	MBZ	
33	BranchCtrl	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	

if - If

32	AtomicCtrl Format: AtomicCtrl										
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".										
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.									
30	Reserved										
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.										
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.	
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields										
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.	
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.										

if - If			
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Illegal

illegal - Illegal			
Source:	Eulsa		
Length Bias:	4		
<p>The Illegal Opcode Exception Enable flag in cr0.1 is normally set so the normal processing of an illegal opcode is to transfer control to the System Routine. Instruction dispatch treats any unused 8-bit opcode (including bit 7 of the instruction, reserved for future opcode expansion) as if it is the illegal opcode. The illegal opcode is zero because that byte value is more likely than most to be read via a wayward instruction pointer. The illegal instruction is an instruction only in the same way that a NULL pointer in software is a pointer. Both are special values indicating invalid instances.</p>			
Format:	illegal		
Restriction			
The illegal instruction takes no instruction options.			
Syntax			
illegal			
Pseudocode			
{ Set the Illegal Opcode Exception Status bit in cr0.1. if (Illegal Opcode Exception Enable is set in cr0.1) { Transfer control to the System Routine (return address to AIP, IP = SIP). }			
Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N
DWord	Bit	Description	
0..3	127:7	Reserved	
		Format:	MBZ
	6:0	Opcode	
		Format:	EU_OPCODE

Join

join - Join			
Source:	Eulsa		
Length Bias:	4		
<p>The join instruction makes the inactive channels active at the join IP if those channels are predicated. Any deactivated channels due to a goto instruction match the join IP are activated (qualified with predicates at join). If no IP is matched at this join, the program goes to the next IP with the active channels which followed the program path up to the join instruction. If no active channels are present after executing the join instruction, the program jumps to the offset specified by JIP instead of next IP. The join instruction is used in conjunction with a goto instruction. The join activates channels that are deactivated by the goto instruction. See the goto instruction for the deactivation rules. The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence. The following table describes the 32-bit JIP. In instruction binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand and is a signed 32-bit number. This value is added to IP pre-increment. If SPF is ON, none of the PcIP are updated.</p>			
Format:	<code>[(pred)] join (exec_size) JIP</code>		
Programming Notes			
An index of 0 is an infinite loop.			
Restriction			
JIP must point to a Flow Control Instruction.			
Syntax			
<code>[(pred)] join (exec_size) imm32</code>			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { // for the predicated channels and the remaining channels PcIP[n] = IP + 1; } } if (PcIP != (IP + 1)) { // for all channels when no channels are activated and no other active channels Jump(IP + JIP); } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N

DWord	Bit	Description
0..3	127:96	Reserved
		Exists If: ([Src0.IsImm]==false)
		Format: MBZ
	127:96	JIP
		Exists If: ([Src0.IsImm]==true)
		Format: S31 The byte-aligned jump distance if a jump is taken for the channel
	95:80	Reserved
		Exists If: ([Src0.IsImm]==false)
		Format: MBZ
	95:64	Reserved
		Exists If: ([Src0.IsImm]==true)
		Format: MBZ
	79:66	Src0.Operand
		Exists If: ([Src0.IsImm]==false)
		Format: DirectOperand
65:64	Reserved	
	Exists If: ([Src0.IsImm]==false)	
	Format: MBZ	
63:50	Dst.Operand	
	Format: DirectOperand	
49:47	Reserved	
	Format: MBZ	
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false
1	true	
45:34	Reserved	
	Format: MBZ	
33	BranchCtrl This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl	
	Format: AtomicCtrl	
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write	

join - Join

enables are used to generate the final write enable. This field should be normally "0".			
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:		MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
	Format:		PredCtrl
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0]		
	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum		

join - Join

		<p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	ChanOff	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff			
18:16	ExecSize	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize			
15:0	Header	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header			

Jump Indexed

jmp - Jump Indexed			
Source:	Eulsa		
Length Bias:	4		
Description			
<p>The jmp instruction redirects program execution to an index offset relative to the pre-incremented instruction pointer. The index is a signed integer value, with positive or zero integers for forward jumps, and negative integers for backward jumps. In instruction binary, index is carried as src0 register or immediate. The ip register must be put (for example, by the assembler) at the dst. Predication is allowed to provide conditional jump with a scalar condition. As the execution size is 1, the first channel of PMASK (flags post prediction control and negate) is used to determine whether the jump is taken or not. If the condition is false, the jump is not taken and execution continues with the next instruction.</p>			
Format:	[(pred)] jmp (1) index {NoMask}		
Programming Notes			
An index argument of 16 would continue to the next instruction (assuming the instruction is encoded as 128b).			
An index argument of 0 loops infinitely: all immediate branch arguments, including jmp now, are relative to the pre-increment IP.			
Restriction			
The execution size must be 1.			
MaskCtrl must be specified.			
QtrCtrl must not be used for jmp instruction.			
Jmp instruction with non-uniform predicates or reg32 source cannot be used when EU Fusion is enabled.			
Syntax			
[(pred)] jmp (1) reg32 {NoMask}		[(pred)] jmp (1) imm32 {NoMask}	
Pseudocode			
Evaluate(WrEn); if (WrEn != 0) { Jump(IP + index); }			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
		Format:	MBZ

jmp*i* - Jump Indexed

127:96	JIP	
	Exists If:	([Src0.IsImm]==true)
	Format:	S31
The byte-aligned jump distance if a jump is taken for the channel		
95:80	Reserved	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
95:64	Reserved	
	Exists If:	([Src0.IsImm]==true)
	Format:	MBZ
79:66	Src0.Operand	
	Exists If:	([Src0.IsImm]==false)
	Format:	DirectOperand
65:64	Reserved	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
63:50	Dst.Operand	
	Format:	DirectOperand
49:47	Reserved	
	Format:	MBZ
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false
	1	true
45:34	Reserved	
	Format:	MBZ
33	BranchCtrl	
	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
		Description

jmp_i - Jump Indexed

	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:		MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
	Format:		PredCtrl
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0]		
	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum		
	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the		

jmpj - Jump Indexed			
	<p>destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

L3_CONTROL

L3_CONTROL - L3_CONTROL			
Source:	BSpec		
Length Bias:	2		
<p>This command provides flexibility to flush selective graphics memory address locations (4KB in size) cached in L3\$ without requiring to flush the complete L3\$. The memory pages to be flushed must be detailed through L3 Flush Address Range record in line to the command. Multiple L3 Flush Address Range records can be programmed through a single L3_CONTROL command.</p> <p>L3 Flush Address Range has number of pages to be flushed and must be a power of two (2^n), this is indicated in terms of number of lower order bits of the address to be masked (AddressMask). L3 Flush Address Range has the starting page of the graphics virtual address to be flushed and must be a power of two (2^m) with a greater value than that of the number of pages ($m \geq n$). Refer L3 Flush Address Range structure for examples.</p> <p>This command is supported by RenderCS and ComputeCS.</p>			
Programming Notes			
<ul style="list-style-type: none"> L3_CONTROL must be always be programmed with at least one "L3 Flush Address Range". On Post-Sync Operation disabled, following the header of the command "L3 Flush Address Range" fields must be programmed skipping the "Address" and "Immediate Data Fields". 			
DWord	Bit	Description	
0	31:29	Type	
		Default Value:	03h GFXPIPE
		Format:	Opcode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	5h L3_CONTROL
		Format:	OpCode
	23	3D Command Sub Opcode	
		Default Value:	1h L3_CONTROL
		Format:	OpCode
22	Reserved		
	Format:	MBZ	
21	Destination Address Type	Defines address space of Destination Address	
	Value	Name	Description
	0h	PPGTT	Use PPGTT address space for DW write

L3_CONTROL - L3_CONTROL		
	1h	GGTT Use GGTT address space for DW write
	Programming Notes	
	Ignored if ""No Write" is selected in Post-Sync Operation.	
20	Command Streamer Stall Enable	
	Format:	U1
	If ENABLED, Command Parser stalls on this command until the command is completely executed.	
	Programming Notes	
19	Reserved	
18:16	Reserved	
	Format:	MBZ
15	Reserved	
	Format:	MBZ
14	Post Sync Operation	
	This field specifies an optional action to be taken upon completion of the synchronization operation.	
	Value	Name Description
	0h	No Write No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
	1h	Write Immediate Data Write the QWord containing Immediate Data Low, High DWs to the Destination Address
13	Reserved	
	Format:	MBZ
12	Reserved	
11	Reserved	
	Format:	MBZ
10	HDC Pipeline Flush	
	Setting this bit ensures HDC pipeline is flushed and the memory transactions are coherent in L3\$ as part of the flush operation prior to triggering of address based range flush to L3	
9	Render Target Cache Flush Enable	
	Setting this bit enables flushing (i.e. writing back the dirty lines to memory and invalidating the tags) of Render Cache (L1) prior to triggering of address based range flush to L3.	
8	Depth Cache Flush	
	Setting this bit enables flushing (i.e. writing back the dirty lines to memory and invalidating the tags) of depth related caches (HiZ cache, Stencil cache and depth cache) prior to triggering of	

		L3_CONTROL - L3_CONTROL			
		address based range flush to L3.			
	7:0	Length			
		Value	Name	Description	Exists If
		5h	[Default]	"Length" field of the command with Post-Snc Operation enabled with 'N' L3 FLush Address Range will be: (2N +3)h	//Post-Sync Operation Enabled, Single L3 Flush Address Range
		1h		"Length" field of the command with Post-Snc Operation disabled with 'N' L3 FLush Address Range will be: (2N -1)h	//Post-Sync Operation Disabled, Single L3 Flush Address Range
1..2	63:0	L3 Flush Address Range			
		Format:		L3_FLUSH_ADDRESS_RANGE	
3..4	63:48	Reserved			
		Format:		MBZ	
	47:3	Address			
		Format:		GraphicsAddress[47:3]	
		Bits 47:3 specify the QW aligned graphics memory address to which the Immediate Qword Data is reported on execution of this command. Ignored if "No Write" is selected in Post-Sync Operation.			
	2:0	Reserved			
		Format:		MBZ	
5..6	63:0	Immediate Data			
		Format:		U64	
		This field specifies the QWord value to be written to the address when Post-Sync Operation is enabled. Only valid when Post-Sync Operation is 1h (Write Immediate Data) and ignored if "No Write" is selected in Post-Sync Operation.			

Leading Zero Detection

lzd - Leading Zero Detection			
Source:	Eulsa		
Length Bias:	4		
The lzd instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is zero, store 32 in dst.			
Format: [(pred)] lzd[.cmod] (exec_size) dst src0			
Syntax			
[(pred)] lzd[.cmod] (exec_size) reg reg [(pred)] lzd[.cmod] (exec_size) reg imm32			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { UD udScalar = src0.chan[n]; UD cnt = 0; while ((udScalar & (1 << 31)) == 0 && cnt != 32) { cnt ++; udScalar = udScalar << 1; } dst.chan[n] = cnt; } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
*B,*W,*D	UD		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0] Exists If: ([Src0.IsImm]==true)	
	95:92	CondCtrl Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: FlagModifier	
	95:64	Src0.ImmValue[63:32] Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))	
	87:84	Src0.VertStride Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: VertStride	
	83:81	Src0.Width Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	

Izd - Leading Zero Detection

		If:	([Src0.DataType]!=:df)
		Format:	Width
80	Src0.AddrMode		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==false)	

Izd - Leading Zero Detection

	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name Description
	0	Normal [Default] Normal. Per channel write enable used for final write enable generation.
	1	NoMask NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name Description
	0	NoCompaction [Default] No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits	

Izd - Leading Zero Detection

	<p>generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		Format:	PredCtrl							
Format:	PredCtrl										
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>										
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>										
21:19	<p>ChanOff</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>		Format:	ChanOff							
Format:	ChanOff										
18:16	<p>ExecSize</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>		Format:	ExecSize							
Format:	ExecSize										
15:0	<p>Header</p> <table border="1"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>		Format:	Header							
Format:	Header										

LO8DS Render Target Write MSD

MSD_RTW_LO8DS - LO8DS Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 0h Format: Opcode Full precision data message
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. <div style="background-color: #e6f2ff; padding: 5px; text-align: center;">Programming Notes</div> This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	Message Type Default Value: 0Ch

MSD_RTW_LO8DS - LO8DS Render Target Write MSD									
	<table border="1"> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table> <p>Render Target Write message</p>	Format:	Opcode						
Format:	Opcode								
13	<p>Per-Sample PS Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.</p> <table border="1"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2"> <p>This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.</p> </td> </tr> </table>	Format:	Enable	Programming Notes		<p>This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.</p>			
Format:	Enable								
Programming Notes									
<p>This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.</p>									
12	<p>Last Render Target Select</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.</p>	Format:	Enable						
Format:	Enable								
11	<p>Slot Group Select</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_RT_SGS</td> </tr> </table> <p>This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.</p>	Format:	MDC_RT_SGS						
Format:	MDC_RT_SGS								
10:8	<p>Render Target Message Subtype</p> <table border="1"> <tr> <td>Default Value:</td> <td>2h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table> <p>SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask.</p> <table border="1"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2"> <p>The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].</p> </td> </tr> </table>	Default Value:	2h	Format:	Opcode	Programming Notes		<p>The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].</p>	
Default Value:	2h								
Format:	Opcode								
Programming Notes									
<p>The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].</p>									
7:0	<p>Binding Table Index</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_BTS</td> </tr> </table> <p>Specifies the Binding Table Index for the message</p>	Format:	MDC_BTS						
Format:	MDC_BTS								

Logic And

and - Logic And			
Source:	Eulsa		
Length Bias:	4		
<p>The and instruction performs component-wise logic AND operation between src0 and src1 and stores the results in dst. Register source operands can use source modifiers: Any source modifier is numeric, optionally changing a source value s to -s, abs(s), or -abs(s) before the AND operation. Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a AND (NOT b) to be calculated with one instruction. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.</p>			
<p>Format:</p> <p style="padding-left: 40px;">Source modifier is not allowed if source is an accumulator.</p>			
Restriction			
Source modifier is not allowed if source is an accumulator.			
Syntax			
<pre>[(pred)] and[.cmod] (exec_size) reg reg reg [(pred)] and[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] & src1.chan[n]; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
	127:96	Src1.ImmValue[31:0]	
	Exists If:	([Src1.IsImm]==true)	
125:122	Reserved		
	Exists If:	([Src1.IsImm]==false)	
	Format:	MBZ	
121:120	Src1.Mod		
	Exists If:	([Src1.IsImm]==false)	

and - Logic And

	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	(([Src1.IsImm]==false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	(([Src1.IsImm]==false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	(([Src1.IsImm]==false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	(([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	(([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	(([Src1.IsImm]==false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	(([Src1.IsImm]==false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	(([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	(([Src0.AddrMode]==Direct)
	Format:	DirectOperand

and - Logic And

79:66	Src0.Operand		
	Exists If:	([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
	65:64	Src0.HorzStride	
		Format:	HorzStride
	63:50	Dst.Operand	
		Exists If:	([Dst.AddrMode]==Direct)
		Format:	DirectOperand
	63:50	Dst.Operand	
		Exists If:	([Dst.AddrMode]==Indirect)
		Format:	IndirectOperand
	49:48	Dst.HorzStride	
		Format:	HorzStride
	47	Src1.IsImm	
		This field indicate that Source 1 operand is carrying an immediate value.	
		Value	Name
		0	false [Default]
		1	true
	46	Src0.IsImm	
This field indicate that Source 0 operand is carrying an immediate value.			
Value		Name	
0		false [Default]	
1		true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==false)	
	Format:	RegDataType	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==true)	
	Format:	ImmDataType	
39:36	Dst.DataType		
	Format:	RegDataType	
35	Dst.AddrMode		
	Format:	AddrMode	
34	Saturate		

and - Logic And

	Format:	Saturate									
33	AccWrCtrl										
	Format:	AccWrCtrl									
32	AtomicCtrl										
	Format:	AtomicCtrl									
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td>1</td> <td>NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.									
30	Reserved										
29	CmptCtrl										
	Format:	MBZ									
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.	
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.	
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	PredCtrl										
	Format:	PredCtrl									

and - Logic And

		This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0]	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff			
18:16	ExecSize	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize			
15:0	Header	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header			

Logic Not

not - Logic Not			
Source:	Eulsa		
Length Bias:	4		
<p>The not instruction performs logical NOT operation (or one's complement) of src0 and storing the results in dst. This operation does not produce sign or overflow conditions. Only the .e/z or .ne/.nz conditional modifiers should be used.</p> <p>A register source operand can use a source modifier: Any source modifier is logical, optionally changing a source value <i>s</i> to $\sim s$ (inverting all source bits). Such a source modifier is not particularly useful with the not instruction, as it changes the effect of not to just copying bits.</p>			
<p>Format:</p> <pre>[(pred)] not[.cmod] (exec_size) dst src0</pre>			
Restriction			
Source modifier is not allowed if source is an accumulator.			
Syntax			
<pre>[(pred)] not[.cmod] (exec_size) reg reg [(pred)] not[.cmod] (exec_size) reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = ~ src0.chan[n]; }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0]	
		Exists If:	([Src0.IsImm]==true)
	95:92	CondCtrl	
		Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))
		Format:	FlagModifier
	95:64	Src0.ImmValue[63:32]	
		Exists If:	([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	Src0.VertStride	
		Exists	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND

not - Logic Not

		If:	([Src0.DataType]!=:df)
		Format:	VertStride
83:81	Src0.Width		
	Exists	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	If:	([Src0.DataType]!=:df)	
	Format:	Width	
80	Src0.AddrMode		
	Exists	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	If:	([Src0.DataType]!=:df)	
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)	
	If:	([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)	
	If:	([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	If:	([Src0.DataType]!=:df)	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists	If:	([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists	If:	([Dst.AddrMode]==Direct)
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	

not - Logic Not

45:44	Src0.Mod	Format:	SrcMod
43:40	Src0.DataType	Exists If:	([Src0.IsImm]==false)
		Format:	RegDataType
43:40	Src0.DataType	Exists If:	([Src0.IsImm]==true)
		Format:	ImmDataType
39:36	Dst.DataType	Format:	RegDataType
35	Dst.AddrMode	Format:	AddrMode
34	Saturate	Format:	Saturate
33	AccWrCtrl	Format:	AccWrCtrl
32	AtomicCtrl	Format:	AtomicCtrl
31	MaskCtrl	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
		Value	Name
		Description	
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.

not - Logic Not

	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	<p>PredInv</p> <p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>			Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>			Format:	PredCtrl							
Format:	PredCtrl											
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>											
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>											
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>			Format:	ChanOff							
Format:	ChanOff											
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>			Format:	ExecSize							
Format:	ExecSize											
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>			Format:	Header							
Format:	Header											

Logic Or

or - Logic Or			
Source:	Eulsa		
Length Bias:	4		
<p>The or instruction performs component-wise logic OR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.</p> <p>Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value <i>s</i> to <i>~s</i> (inverting all source bits). This capability allows expressions like a OR (NOT b) to be calculated with one instruction.</p> <p>Format:</p> <pre>[(pred)] or[.cmod] (exec_size) dst src0 src1</pre>			
Restriction			
Source modifier is not allowed if source is an accumulator.			
Syntax			
<pre>[(pred)] or[.cmod] (exec_size) reg reg reg [(pred)] or[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] src1.chan[n]; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
125:122		Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
121:120		Src1.Mod	
		Exists If:	([Src1.IsImm]==false)

or - Logic Or

	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	((Src1.IsImm) == false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	((Src1.IsImm) == false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	((Src0.AddrMode) == Direct)
	Format:	DirectOperand

or - Logic Or

79:66	Src0.Operand	Exists If:	([Src0.AddrMode]==Indirect)
		Format:	IndirectOperand
65:64	Src0.HorzStride	Format:	HorzStride
63:50	Dst.Operand	Exists If:	([Dst.AddrMode]==Direct)
		Format:	DirectOperand
63:50	Dst.Operand	Exists If:	([Dst.AddrMode]==Indirect)
		Format:	IndirectOperand
49:48	Dst.HorzStride	Format:	HorzStride
47	Src1.IsImm	This field indicate that Source 1 operand is carrying an immediate value.	
		Value	Name
		0	false [Default]
		1	true
46	Src0.IsImm	This field indicate that Source 0 operand is carrying an immediate value.	
		Value	Name
		0	false [Default]
		1	true
45:44	Src0.Mod	Format:	SrcMod
43:40	Src0.DataType	Exists If:	([Src0.IsImm]==false)
		Format:	RegDataType
43:40	Src0.DataType	Exists If:	([Src0.IsImm]==true)
		Format:	ImmDataType
39:36	Dst.DataType	Format:	RegDataType
35	Dst.AddrMode	Format:	AddrMode
34	Saturate		

or - Logic Or

	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
		Description
	0	Normal [Default]
		Normal. Per channel write enable used for final write enable generation.
	1	NoMask
		NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name
		Description
	0	NoCompaction [Default]
		No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted
		Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name
		Description
	0	Positive [Default]
		Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative
		Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl	
	Format:	PredCtrl

or - Logic Or

		This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.			
	23	FlagRegNum[0]	This field specifies bit[0] of the register number for a flag register operand.		
	22	FlagSubRegNum	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
	21:19	ChanOff	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff				
	18:16	ExecSize	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize				
	15:0	Header	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header				

Logic Xor

xor - Logic Xor			
Source:	Eulsa		
Length Bias:	4		
<p>The xor instruction performs component-wise logic XOR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.</p> <p>Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value <i>s</i> to <i>~s</i> (inverting all source bits). This capability allows expressions like a XOR (NOT b) to be calculated with one instruction.</p> <p>Format:</p> <pre>[(pred)] xor[.cmod] (exec_size) dst src0 src1</pre>			
Restriction			
Source modifier is not allowed if source is an accumulator.			
Syntax			
<pre>[(pred)] xor[.cmod] (exec_size) reg reg reg [(pred)] xor[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] ^ src1.chan[n]; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
	Exists If:	([Src1.IsImm]==true)	
125:122	125:122	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
121:120		Src1.Mod	
	Exists If:	([Src1.IsImm]==false)	

xor - Logic Xor

	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	((Src1.IsImm) == false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	((Src1.IsImm) == false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	((Src0.AddrMode) == Direct)
	Format:	DirectOperand

xor - Logic Xor

79:66	Src0.Operand		
	Exists If:	([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
	65:64	Src0.HorzStride	
		Format:	HorzStride
	63:50	Dst.Operand	
		Exists If:	([Dst.AddrMode]==Direct)
		Format:	DirectOperand
	63:50	Dst.Operand	
		Exists If:	([Dst.AddrMode]==Indirect)
		Format:	IndirectOperand
	49:48	Dst.HorzStride	
		Format:	HorzStride
	47	Src1.IsImm	
		This field indicate that Source 1 operand is carrying an immediate value.	
		Value	Name
		0	false [Default]
	1	true	
	46	Src0.IsImm	
This field indicate that Source 0 operand is carrying an immediate value.			
Value		Name	
0		false [Default]	
1	true		
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==false)	
	Format:	RegDataType	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==true)	
	Format:	ImmDataType	
39:36	Dst.DataType		
	Format:	RegDataType	
35	Dst.AddrMode		
	Format:	AddrMode	
34	Saturate		

xor - Logic Xor

		Format:	Saturate
33	AccWrCtrl		
	Format:	AccWrCtrl	
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:	MBZ	
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
	Format:	PredCtrl	

xor - Logic Xor			
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">ChanOff</td> </tr> </table> This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.	Format:	ChanOff
Format:	ChanOff		
18:16	ExecSize <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">ExecSize</td> </tr> </table> This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.	Format:	ExecSize
Format:	ExecSize		
15:0	Header <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

MEDIA_CURBE_LOAD

MEDIA_CURBE_LOAD						
Source:		RenderCS				
Length Bias:		2				
DWord	Bit	Description				
0	31:29	Command Type				
		Default Value:	3h GFXPIPE			
		Format:	OpCode			
	28:27	Pipeline				
		Default Value:	2h Media			
		Format:	OpCode			
	26:24	Media Command Opcode				
		Default Value:	0h MEDIA_CURBE_LOAD			
		Format:	OpCode			
	23:16	SubOpcode				
Default Value:		1h MEDIA_CURBE_LOAD SubOp				
Format:		OpCode				
15:0	DWord Length					
	Format:	=n				
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>2h</td> <td>DWORD_COUNT_n [Default]</td> <td>Excludes DWord (0,1)</td> </tr> </tbody> </table>	Value	Name	Description	2h	DWORD_COUNT_n [Default]
Value	Name	Description				
2h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)				
1	31:0	Reserved				
		Format:	MBZ			
2	31:17	Reserved				
		Format:	MBZ			
	16:0	CURBE Total Data Length				
		Format:	U17			
		<p>This field provides the length in bytes of the CURBE data. This field must have the same alignment as the Curbe Object Data Start Address. As the CURBE data are sent directly to ROB, range is limited to CURBE Allocation Size.</p> <p>This field must be 64-byte aligned.</p>				
3	31:0	CURBE Data Start Address				
		Format:	DynamicStateOffset[31:0]			
		Specifies the 64-byte aligned address of the CURBE data. This pointer is relative to the Dynamics Base Address .				

MEDIA_CURBE_LOAD			
		Value	Name
		[0, FFFFFFFFh]	

MEDIA_INTERFACE_DESCRIPTOR_LOAD

MEDIA_INTERFACE_DESCRIPTOR_LOAD			
Source:	RenderCS		
Length Bias:	2		
A Media_State_Flush should be used before this command to ensure that the temporary Interface Descriptor storage is cleared.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	0h MEDIA_INTERFACE_DESCRIPTOR_LOAD
		Format:	OpCode
	23:16	SubOpcode	
Default Value:		2h MEDIA_INTERFACE_DESCRIPTOR_LOAD SubOp	
Format:		OpCode	
15:0	DWord Length	Format:	=n
	Value	Name	Description
	2h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)
1	31:0	Reserved	
		Format:	MBZ
2	31:17	Reserved	
		Format:	MBZ
	16:0	Interface Descriptor Total Length	
		Format:	U17
		This field provides the length in bytes of the Interface Descriptor data. This field must have the same alignment as the Interface Descriptor Data Start Address. It must be DQWord (32-byte) aligned. As the Interface Descriptor data are sent directly to ROB, range is limited to CURBE Allocation Size.	
Value	Name		
[32,2048]	[1,64] interface descriptor entries		
3	31:0	Interface Descriptor Data Start Address	

MEDIA_INTERFACE_DESCRIPTOR_LOAD

		Format: DynamicStateOffset[31:0]INTERFACE_DESCRIPTOR_DATA	
		This bit specifies the <u>64-byte</u> aligned address of the Interface Descriptor data. This pointer is relative to the Dynamics Base Address.	
		Value	Name
		[0, FFFFFFFFh]	



MEDIA_OBJECT_GRPID

MEDIA_OBJECT_GRPID			
Source:	RenderCS		
Length Bias:	2		
<p>The MEDIA_OBJECT_GRPID command is a variation of MEDIA_OBJECT which includes a group id which is used to allocate and track Barriers and Shared Local Memory. The Interface Descriptor is used to specify how much SLM is needed and how many threads will be reporting to the Barrier. All MEDIA_OBJECT_GRPIDs with the same group id should have the same interface descriptor and be dispatched to the same Tslice the dispatcher will ensure this if Force Destination = 0, but software must ensure this if Force Destination = 1. Software should also ensure that all the threads needed for the Barrier will fit into a Tslice, or the Barrier will never be satisfied. Either SLM or a barrier must be used with MEDIA_OBJECT_GRPID, if neither is needed then a MEDIA_OBJECT must be used instead.</p> <p>MEDIA_OBJECT_GRPID supports the GPGPU version of payload delivery either indirect or CURBE can be split between the threads in a group (per-thread payload), as well as a section which is sent to all threads (cross-thread payload). See the GPGPU payload section. For indirect, the same pointer must be sent with all the commands associated with the thread group for payload splitting to work properly. Inline data is not split, but the payload attached to each command is sent with that thread. Only one of inline, indirect, or CURBE is allowed, but at least one form of payload must be sent.</p> <p>MEDIA_STATE_FLUSH with the watermark bit must be placed between groups created by MEDIA_OBJECT_GRPID. The Interface Descriptor associated with the watermark must match the Interface Descriptor used for the following group.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Media Command Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	1h MEDIA_OBJECT_GRPID
		Format:	OpCode
	23:16	Media Command Sub-Opcode	
		Default Value:	6h MEDIA_OBJECT_GRPID SubOp
		Format:	OpCode
	15:0	DWord Length	
		Format:	=n
	<p>Excludes DWords 0,1 Generic Mode: DWord Length = N+5, where N is in the range of [0,504]. The maximum is 504 DW (equivalent to 63 8-DW registers). When both inline and indirect data are fetched for this command, the total size in 8-DW registers must be less than 112 (with both inline data length N and indirect data length rounded up to 8-DW aligned individually). The</p>		

MEDIA_OBJECT_GRPID								
		<p>minimal inline data length is 0.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>5h</td> <td>DWORD_COUNT_n [Default]</td> </tr> <tr> <td>[5,112]</td> <td>Min_Max</td> </tr> </tbody> </table>	Value	Name	5h	DWORD_COUNT_n [Default]	[5,112]	Min_Max
Value	Name							
5h	DWORD_COUNT_n [Default]							
[5,112]	Min_Max							
1	31:8	Reserved						
	7:6	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
5:0	Interface Descriptor Offset <table border="1"> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,30]</td> <td></td> </tr> </tbody> </table>	Format:	U6	Value	Name	[0,30]		
Format:	U6							
Value	Name							
[0,30]								
2	31:24	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	23	End of Thread Group This bit indicates that this dispatch is the last for the current thread group.						
	22:17	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
16:0	Indirect Data Length <table border="1"> <tr> <td>Format:</td> <td>U17</td> </tr> </table> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to URB, range is limited to(URB Entry Allocation Size)*(Number of URB Entries) in the MEDIA_VFE_STATE command.</p>	Format:	U17					
Format:	U17							
3	31:0	Indirect Data Start Address <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:0]</td> </tr> </table>	Format:	GraphicsAddress[31:0]				
		Format:	GraphicsAddress[31:0]					
		<p>This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the Indirect Object Base Address. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation.</p> <p>It is the 64-byte aligned address of the indirect data.</p>						
<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>[0-512MB]</td> <td></td> <td>Bits 31:29 MBZ</td> </tr> </tbody> </table>	Value	Name	Description	[0-512MB]		Bits 31:29 MBZ		
Value	Name	Description						
[0-512MB]		Bits 31:29 MBZ						
4	31:25	Reserved						

MEDIA_OBJECT_GRPID		
		Format: MBZ
	24:16	Y Position Format: U9 This field provides the Y position of the block to be dispatched. It is sent the kernel via the R0 header, but is not otherwise used by hardware.
	15:9	Reserved Format: MBZ
	8:0	X Position Format: U9 This field provides the X position of the block to be dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.
5	31:24	Reserved Format: MBZ
	23:16	Block Color Format: U8 This field specifies the color value associated with the block to be dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.
	15:0	Reserved Format: MBZ
6	31:0	GroupID A unique identifying number which describes the threads which share a barrier and/or SLM. Reuse of numbers is allowed as long as the old group is not currently running.
7..n	31:0	Inline Data The format of this data is specified by software. Hardware does not interpret this data; it merely passes it to the kernel for processing. The total size for the inline data and indirect data must not exceed 112 registers.

MEDIA_OBJECT

MEDIA_OBJECT			
Source:	RenderCS		
Length Bias:	2		
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Media Command Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	1h MEDIA_OBJECT
		Format:	OpCode
	23:16	Media Command Sub-Opcode	
Default Value:		0h MEDIA_OBJECT SubOp	
Format:		OpCode	
15	Reserved		
	Format:	MBZ	
14:0	DWord Length		
	Default Value:	[4,116] Min_Max	
	Format:	=n	
Excludes DWords 0,1 Generic Mode: DWord Length = N+4, where N is in the range of [0,504]. The maximum is 504 DW (equivalent to 63 8-DW registers). When both inline and indirect data are fetched for this command, the total size in 8-DW registers must be less than 112 (with both inline data length N and indirect data length rounded up to 8-DW aligned individually). The minimal inline data length is 0.			
1	31:8	Reserved	
	7:6	Reserved	
		Format:	MBZ
5:0	Interface Descriptor Offset		
Format:		U6	
This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.			
2	31:27	Reserved	

MEDIA_OBJECT												
	<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
26:25	<p>Slice Destination Select MSBs These bits are the MSBs of the slice destination select field. The LSB bits are 20:19. Only slices that are available can be selected.</p>											
24	<p>Thread Synchronization This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No thread synchronization</td> </tr> <tr> <td>1</td> <td>Thread dispatch is synchronized by the 'spawn root thread' message</td> </tr> </tbody> </table>	Value	Name	0	No thread synchronization	1	Thread dispatch is synchronized by the 'spawn root thread' message					
Value	Name											
0	No thread synchronization											
1	Thread dispatch is synchronized by the 'spawn root thread' message											
23	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
22	<p>Force Destination If set, bits 20:17 are used to determine the destination of this dispatch, if clear the destination will be chosen based on load.</p>											
21	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
20:17	<p>Subslice Destination Select</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U4</td> </tr> </table> <p>This field selects the SubSlice that this thread must be sent to. This configuration has only 1 slice. Ignored if Force Destination = 0.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 65%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0-7]</td> <td>Subslice ID</td> <td>Valid subslice number in this configuration.</td> </tr> <tr> <td>[8-15]</td> <td>Reserved</td> <td>Illegal values. Results undefined if used.</td> </tr> </tbody> </table>	Format:	U4	Value	Name	Description	[0-7]	Subslice ID	Valid subslice number in this configuration.	[8-15]	Reserved	Illegal values. Results undefined if used.
Format:	U4											
Value	Name	Description										
[0-7]	Subslice ID	Valid subslice number in this configuration.										
[8-15]	Reserved	Illegal values. Results undefined if used.										
16:0	<p>Indirect Data Length</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U17</td> </tr> </table> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to URB, range is limited to (URB Entry Allocation Size)*(Number of URB Entries) in the MEDIA_VFE_STATE command. When both inline and indirect data are fetched for this command, enough room should be allowed for at least 1 copy of the indirect and 1 copy of the inline data in the URB.</p>	Format:	U17									
Format:	U17											
3	<p>31:0 Indirect Data Start Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Format:</td> <td style="width: 60%;">GraphicsAddress[31:0]</td> </tr> </table> <p>This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the Indirect Object Base Address. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode</p>	Format:	GraphicsAddress[31:0]									
Format:	GraphicsAddress[31:0]											

MEDIA_OBJECT					
	<p>of operation.</p> <p>This field specifies the 64-byte aligned address of the indirect data.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,512MB]</td> <td></td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Bits 31:29 MBZ</p>	Value	Name	[0,512MB]	
Value	Name				
[0,512MB]					
4	<p>31:25 Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ			
	<p>24:16 Y Position</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">U9</td> </tr> </table> <p>This field provides the Y position of the block dispatched by this command. It is sent to the kernel via the R0 header but is not otherwise used by hardware.</p>	Format:	U9		
	Format:	U9			
<p>15:9 Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ				
<p>8:0 X Position</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">U9</td> </tr> </table> <p>This field provides the X position of the block dispatched by this command. It is sent to the kernel via the R0 header but is not otherwise used by hardware.</p>	Format:	U9			
Format:	U9				
5	<p>31:24 Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ			
	<p>23:16 Block Color</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">U8</td> </tr> </table> <p>This field specifies the color for the block being dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.</p>	Format:	U8		
Format:	U8				
<p>15:0 Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ				
6..n	<p>31:0 Inline Data</p> <p>Generic Mode: The format of this data is specified by software. Hardware does not interpret this data; it merely passes it to the kernel for processing. The total size for the inline data and indirect data must not exceed 112 registers.</p>				

MEDIA_OBJECT_WALKER

MEDIA_OBJECT_WALKER			
Source:		RenderCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	1h MEDIA_OBJECT_WALKER
		Format:	OpCode
	23:16	SubOpcode	
Default Value:		03h MEDIA_OBJECT_WALKER SubOp	
Format:		OpCode	
15	Reserved		
	Format:	MBZ	
14:0	DWord Length	Default Value:	[15,112] Min_Max
		Format:	=n
	Note: If this field is greater than 15, it indicates that inline data is present. If present, inline data is common for all threads generated from this command, If this field is 15, it indicates that inline data is not present. It should be noted that unlike other media object command, inline data is optional for this command.		
1	31:8	Reserved	
	7:6	Reserved	
		Format:	MBZ
5:0	Interface Descriptor Offset		
	Format:	U6	
This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.			
2	31:25	Reserved	
		Format:	MBZ

MEDIA_OBJECT_WALKER

	24	<p>Thread Synchronization</p> <p>This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No thread synchronization</td> </tr> <tr> <td>1</td> <td>Thread dispatch is synchronized by the 'spawn root thread' message</td> </tr> </tbody> </table>	Value	Name	0	No thread synchronization	1	Thread dispatch is synchronized by the 'spawn root thread' message								
	Value	Name														
	0	No thread synchronization														
	1	Thread dispatch is synchronized by the 'spawn root thread' message														
	23:22	<p>Masked Dispatch</p> <p>Format: U2</p> <p>Enable the masking of the dispatch of individual threads based on a bitmask read from CURBE, and specifies the pitch of the CURBE surface. If enabled, CURBE will not be used for thread payload.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td></td> <td>Masked Dispatch Disabled</td> </tr> <tr> <td>01b</td> <td></td> <td>Masked Dispatch with 128-bit pitch in CURBE</td> </tr> <tr> <td>10b</td> <td></td> <td>Masked Dispatch with 256-bit pitch in CURBE</td> </tr> <tr> <td>11b</td> <td></td> <td>Masked Dispatch with 512-bit pitch in CURBE</td> </tr> </tbody> </table>	Value	Name	Description	00b		Masked Dispatch Disabled	01b		Masked Dispatch with 128-bit pitch in CURBE	10b		Masked Dispatch with 256-bit pitch in CURBE	11b	
Value	Name	Description														
00b		Masked Dispatch Disabled														
01b		Masked Dispatch with 128-bit pitch in CURBE														
10b		Masked Dispatch with 256-bit pitch in CURBE														
11b		Masked Dispatch with 512-bit pitch in CURBE														
21:17	<p>Reserved</p> <p>Format: MBZ</p>															
16:0	<p>Indirect Data Length</p> <p>Format: U17</p> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to a URB entry, the range is limited to the URB Entry Allocation Size specified in the MEDIA_VFE_STATE command.</p>															
3	<p>31:0 Indirect Data Start Address</p> <p>This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the Indirect Object Base Address. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation.</p> <p>It is the 64-byte aligned address of the indirect data</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[0 - 512MB]</td> <td></td> <td>(Bits 31:29 MBZ)</td> </tr> </tbody> </table>	Value	Name	Description	[0 - 512MB]		(Bits 31:29 MBZ)									
Value	Name	Description														
[0 - 512MB]		(Bits 31:29 MBZ)														
4	<p>31:0 Reserved</p> <p>Format: MBZ</p>															
5	<p>31:8 Group ID Loop Select</p> <p>This bit field chooses which of the nested loops of the walker are used to identify threads which share a group id and therefore a shared barrier and SLM. The programmer must ensure that each</p>															

MEDIA_OBJECT_WALKER

group will fit into a single subslice. When barriers are enabled every group must have the same number of threads matching the number specified in the Interface Descriptor.

Value	Name	Description
0	No_Groups	Groups are not created, barriers and SLM are not allocated
2	InnerLocal_Groups	Each complete iteration of the Inner local loop and Color loop defines a group, the group id is the concatenation of the Outer global loop to the Mid local loop execution counts.
3	MidLocal_Groups	Each complete iteration of the Mid local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop to the Outer local loop execution counts.
4	OuterLocal_Groups	Each complete iteration of the Outer local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop and the Inner global loop execution counts.
5	InnerGlobal_Groups	Each complete iteration of the Inner global loop and lower loops defines a group, the group id is the Outer global loop execution count.
Others	Reserved	

Restriction

When media thread groups are formed (i.e. Group ID Loop Select is non-zero), software must ensure the size of the group does not exceed the maximum number of threads in a DSS (112).

7:0 Partition Count

Format:	U8-1
---------	------

This field specifies the number of partitions (minus one) used to select thread destinations. The walk position **Color** modulo the **Partition Count** chooses a non-overlapping set of DSS that the dispatch is directed to.

This field is ignored if **MEDIA_VFE_STATE Dispatch Load Balance** is not set to Color mode.

Value	Name
[0-7]	Supported Partition Counts

Programming Notes

For most efficient use of all the DSS, the **Partition Count** should be an even factor of the HW configuration's number of DSS. Also, the **Color Count** should be a multiple of the **Partition Count**.

6 31:24 Color Count Minus One

Format:	U8
---------	----

This field specifies the number of repeat of the inner most loop of the walker. Each repeated walk position is assigned with an incremental Color number. The Color number together with the X and Y position of the thread is used for dependency scoreboard control. Usage Example: This allows multiple sets of dependency threads to be dispatched.

MEDIA_OBJECT_WALKER		
	23:21	Reserved Format: MBZ
	20:16	Middle Loop Extra Steps Format: U5
	15:14	Reserved Format: MBZ
	13:12	Local Mid-Loop Unit Y Format: S1
	11:10	Reserved Format: MBZ
	9:8	Mid-Loop Unit X Format: S1
	7:0	Reserved Format: MBZ
7	31:28	Reserved Format: MBZ
	27:16	Global Loop Exec Count Format: U12
	15:12	Reserved Format: MBZ
	11:0	Local Loop Exec Count Format: U12
8	31:27	Reserved Format: MBZ
	26:16	Block Resolution Y Format: U11 Vertical resolution of the local loop.
	15:11	Reserved Format: MBZ
	10:0	Block Resolution X Format: U11 Horizontal resolution of the local loop.
9	31:27	Reserved Format: MBZ
	26:16	Local Start Y

MEDIA_OBJECT_WALKER		
		Format: U11 Starting vertical position of the local loop.
	15:11	Reserved Format: MBZ
	10:0	Local Start X Format: U11 Starting horizontal position of the local loop.
10	31:0	Reserved Format: MBZ
11	31:28	Reserved Format: MBZ
	27:16	Local Outer Loop Stride Y Format: S11 Vertical stride of the local outer loop, in 2's complement.
	15:12	Reserved Format: MBZ
	11:0	Local Outer Loop Stride X Format: S11 Horizontal stride of the local outer loop, in 2's complement.
12	31:28	Reserved Format: MBZ
	27:16	Local Inner Loop Unit Y Format: S11 Vertical stride of the local inner loop, in 2's complement.
	15:12	Reserved Format: MBZ
	11:0	Local Inner Loop Unit X Format: S11 Horizontal stride of the local inner loop, in 2's complement.
13	31:27	Reserved Format: MBZ
	26:16	Global Resolution Y

MEDIA_OBJECT_WALKER		
		Format: U11 Vertical resolution of the global loop.
	15:11	Reserved Format: MBZ
	10:0	Global Resolution X Format: U11 Horizontal resolution of the global loop.
14	31:28	Reserved Format: MBZ
	27:16	Global Start Y Format: S11 Starting vertical location of the global loop, in 2's complement.
	15:12	Reserved Format: MBZ
	11:0	Global Start X Format: S11 Starting horizontal location of the global loop, in 2's complement.
15	31:28	Reserved Format: MBZ
	27:16	Global Outer Loop Stride Y Format: S11 Vertical stride of the global outer loop, in 2's complement.
	15:12	Reserved Format: MBZ
	11:0	Global Outer Loop Stride X Format: S11 Horizontal stride of the global outer loop, in 2's complement.
16	31:28	Reserved Format: MBZ
	27:16	Global Inner Loop Unit Y Format: S11 Vertical stride of the global inner loop, in 2's complement.
	15:12	Reserved Format: MBZ
	11:0	Global Inner Loop Unit X Format: S11

MEDIA_OBJECT_WALKER		
		Horizontal stride of the global inner loop, in 2's complement.
17..n	31:0	Inline Data

MEDIA_STATE_FLUSH

MEDIA_STATE_FLUSH			
Source:	RenderCS		
Length Bias:	2		
<p>This command updates the Message Gateway state. In particular, it updates the state for a selected Interface Descriptor.</p> <p>This command can be considered same as a MI_Flush except that only media parser will get flushed instead of the entire 3D/media render pipeline. The command should be programmed prior to new Media state, curbe and/or interface descriptor commands when switching to a new context or programming new state for the same context. With this command, pipelined state change is allowed for the media pipe.</p> <p>Be cautious when using this command when child_present flag in the media state is enabled. This is because that CURBE state as well as Interface Descriptor state are shared between root threads and child threads. Changing these states while child threads are generated on the fly may cause unexpected behavior. Combining with MI_ARB_ON/OFF command, it is possible to support interruptibility with the following command sequence where interrupt may be allowed only when MI_ARB_ON_OFF is ON:</p> <pre>MEDIA_STATE_FLUSH VFE_STATE // VFE will hold CS if watermark isn't met MI_ARB_OFF // There must be at least one VFE command before this one MEDIA_OBJECT ... MI_ARB_ON</pre>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	0h MEDIA_STATE_FLUSH
		Format:	OpCode
	23:16	SubOpcode	
Default Value:		4h MEDIA_STATE_FLUSH SubOp	
Format:		OpCode	
15:0	DWord Length		
	Format:	=n	
	Value	Name	Description
	0h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)
1	31:8	Reserved	
		Format:	MBZ
	7	Flush to GO	

MEDIA_STATE_FLUSH			
	<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This bit indicates that the write data out of this thread group should be flushed to the point where it is visible to following commands.</p>	Format:	Enable
Format:	Enable		
6	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
5:0	<p>Interface Descriptor Offset</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U6</td> </tr> </table> <p>This field specifies the offset from the interface descriptor base pointer to the interface descriptor which describes what resources are required to meet the watermark.</p>	Format:	U6
Format:	U6		

MEDIA_VFE_STATE

MEDIA_VFE_STATE						
Source:	RenderCS					
Length Bias:	2					
<p>A stalling PIPE_CONTROL is required before MEDIA_VFE_STATE unless the only bits that are changed are scoreboard related: Scoreboard Enable, Scoreboard Type, Scoreboard Mask, Scoreboard * Delta. For these scoreboard related states, a MEDIA_STATE_FLUSH is sufficient.</p> <ul style="list-style-type: none"> MEDIA_STATE_FLUSH (optional, only if barrier dependency is needed) MEDIA_INTERFACE_DESCRIPTOR_LOAD (optional) 						
DWord	Bit	Description				
0	31:29	Command Type				
		Default Value: 3h GFXPIPE				
	Format: OpCode					
	28:27	Pipeline				
		Default Value: 2h Media				
	Format: OpCode					
26:24	Media Command Opcode					
	Default Value: 0h MEDIA_VFE_STATE					
Format: OpCode						
23:16	SubOpcode					
	Default Value: 0h MEDIA_VFE_STATE SubOp					
Format: OpCode						
15:0	DWord Length					
	Format: =n					
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>07h</td> <td>DWORD_COUNT_n [Default]</td> <td>Excludes DWord (0,1)</td> </tr> </tbody> </table>	Value	Name	Description	07h	DWORD_COUNT_n [Default]
Value	Name	Description				
07h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)				
1	31:10	Scratch Space Base Pointer				
		Format: GeneralStateOffset[31:10]				
	Specifies the 1k-byte aligned address offset to scratch space for use by the kernel. This pointer is relative to the General State Base Address .					
9:8	Reserved					
Format: MBZ						
7:4	Stack Size					

MEDIA_VFE_STATE								
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,11]</td> <td></td> <td>indicating [1KBytes, 2MBytes]</td> </tr> </tbody> </table>	Value	Name	Description	[0,11]		indicating [1KBytes, 2MBytes]
Value	Name	Description						
[0,11]		indicating [1KBytes, 2MBytes]						
		Programming Notes						
		Since the stack uses the upper portion of the scratch space, Stack Size = < Per Thread Scratch Space						
	3:0	Per Thread Scratch Space Format: U4 Specifies the amount of scratch space allowed to be used by each thread. <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,11]</td> <td></td> <td>indicating [1k bytes, 2 Mbytes]: 0=1k, 1=2k, 2=4k, 3=8k 11=2M</td> </tr> </tbody> </table>	Value	Name	Description	[0,11]		indicating [1k bytes, 2 Mbytes]: 0=1k, 1=2k, 2=4k, 3=8k 11=2M
Value	Name	Description						
[0,11]		indicating [1k bytes, 2 Mbytes]: 0=1k, 1=2k, 2=4k, 3=8k 11=2M						
		Programming Notes						
		The driver must allocate enough contiguous scratch space, pointed to by the Scratch Space Pointer, to ensure that all threads have their own per-thread scratch space. Scratch Space Size = (MEDIA_VFE_STATE::Per Thread Scratch Space) * (MEDIA_VFE_STATE::Maximum Number Of Threads). If a fused configuration has fewer threads than the native POR configuration, the scratch space allocation is based on the number of threads in the base native POR configuration.						
2	31:16	Reserved Format: MBZ						
	15:0	Scratch Space Base Pointer High Format: GeneralStateOffset[47:32] This field specifies the high 16 bits of starting address of the Scratch Space Base Pointer						
3	31:16	Maximum Number of Threads Format: U16-1 <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 100%; text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td> Range: [0, 2¹⁶-1], representing [1, 2¹⁶] threads. Normally set to the maximum number of threads: (# EUs) * (# threads/EU). See <i>Graphics Processing Engine</i> for listing of #EUs and #threads in each device. See Programming Restrictions here for additional limitations. Restriction : The smallest number of maximum threads supported is 64. The largest number may exceed the number of threads on the GPU, but must be <= (#Slices)*1024. (Range of FFTID per slice is 10 bits.) </td> </tr> </tbody> </table>	Description	Range: [0, 2 ¹⁶ -1], representing [1, 2 ¹⁶] threads. Normally set to the maximum number of threads: (# EUs) * (# threads/EU). See <i>Graphics Processing Engine</i> for listing of #EUs and #threads in each device. See Programming Restrictions here for additional limitations. Restriction : The smallest number of maximum threads supported is 64. The largest number may exceed the number of threads on the GPU, but must be <= (#Slices)*1024. (Range of FFTID per slice is 10 bits.)				
Description								
Range: [0, 2 ¹⁶ -1], representing [1, 2 ¹⁶] threads. Normally set to the maximum number of threads: (# EUs) * (# threads/EU). See <i>Graphics Processing Engine</i> for listing of #EUs and #threads in each device. See Programming Restrictions here for additional limitations. Restriction : The smallest number of maximum threads supported is 64. The largest number may exceed the number of threads on the GPU, but must be <= (#Slices)*1024. (Range of FFTID per slice is 10 bits.)								
		Programming Notes						
		MSB will be zero due to the range limit below.						

MEDIA_VFE_STATE										
	15:8	<p>Number of URB Entries</p> <table border="1"> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>Specifies the number of URB entries that are used by the unit.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[1,128]</td> <td></td> <td>[1,128] Entries</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Please note that 0 is not allowed for this field.</p>	Format:	U8	Value	Name	Description	[1,128]		[1,128] Entries
	Format:	U8								
	Value	Name	Description							
	[1,128]		[1,128] Entries							
	7	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									
6	<p>Fused EU Dispatch</p> <table border="1"> <tr> <td>Format:</td> <td>Disable</td> </tr> </table> <p>This field determine if threads will be dispatched in sets to fused EUs if set or if they will be dispatched individually. Depending of the project the set size can be 2 or 4. If dispatched in sets the fused threads will all be part of the same thread group for GPGPU threads or will be part of the same iteration of the inner local loop if media threads.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>1h</td> <td>Legacy Mode, threads are not fused</td> </tr> <tr> <td>0h</td> <td>Fused EU Mode</td> </tr> </tbody> </table>	Format:	Disable	Value	Name	1h	Legacy Mode, threads are not fused	0h	Fused EU Mode	
Format:	Disable									
Value	Name									
1h	Legacy Mode, threads are not fused									
0h	Fused EU Mode									
5:3	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
2	<p>Dispatch Load Balance</p> <p>This bit determines how media threads are dispatched between the various dual subslices. GPGPU threads are not impacted by this bit.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Color</td> <td>When this value is used the threads are split into groups depending on the color value for media threads. The number of groups is specified by MEDIA_OBJECT_WALKER Partition Count. Each group is sent to non-overlapping dual subslices. The least loaded active dual subslice available for the group will be selected for all the threads in that group. This allows color to be used to separate workloads with different operations to get better cache coherency.</td> </tr> <tr> <td>0</td> <td>Least Loaded</td> <td>When this value is used the threads are sent to the least loaded dual subslice of all active dual subslices. If media with groups is being used then each group is kept in the same dual subslice.</td> </tr> </tbody> </table>	Value	Name	Description	1	Color	When this value is used the threads are split into groups depending on the color value for media threads. The number of groups is specified by MEDIA_OBJECT_WALKER Partition Count . Each group is sent to non-overlapping dual subslices. The least loaded active dual subslice available for the group will be selected for all the threads in that group. This allows color to be used to separate workloads with different operations to get better cache coherency.	0	Least Loaded	When this value is used the threads are sent to the least loaded dual subslice of all active dual subslices. If media with groups is being used then each group is kept in the same dual subslice.
Value	Name	Description								
1	Color	When this value is used the threads are split into groups depending on the color value for media threads. The number of groups is specified by MEDIA_OBJECT_WALKER Partition Count . Each group is sent to non-overlapping dual subslices. The least loaded active dual subslice available for the group will be selected for all the threads in that group. This allows color to be used to separate workloads with different operations to get better cache coherency.								
0	Least Loaded	When this value is used the threads are sent to the least loaded dual subslice of all active dual subslices. If media with groups is being used then each group is kept in the same dual subslice.								
	1:0	Reserved								
4	31:8	Reserved								
	7:0	<p>Maximum Number of Dual-Subslices</p> <p>This field determines the maximum number of dual subslices which are allowed to be used. Lowering this value below the maximum number of dual subslices available will reduce the</p>								

MEDIA_VFE_STATE		
		memory footprint for context and scratch space, as well as potentially saving power. A zero in this field indicates that all available dual subslices should be enabled. Setting this field to a value larger or equal to the available dual subslices will also enable all of them.
5	31:16	URB Entry Allocation Size
		Format: U16
		Specifies the length of each URB entry used by the unit, in 256-bit (32 byte) register increments. Each URB entry is rounded up to be 64-byte aligned. ROB address for URB starts after CURBE Allocated region.
		Programming Notes
		When Inline data is used with MEDIA_OBJECT or MEDIA_OBJECT_WALKER, then the URB entry allocation size must match the Inline data size. If Indirect data is being used with MEDIA_OBJECT or GPGPU_WALKER then the allocation size must be sufficient for the Indirect data. If both Inline and Indirect are being used, then the allocation size must match the sum of the Inline and Indirect.
		The total size of URB used by VFE must be less than number of bytes allocated for the URB in the L3 banks (value in L3CNTLREG / 32 bytes per entry). The total size of URB used by VFE = Number of Interface Descriptors (=64) + CURBE Allocation Size + (Number of URB Entries * (roundup(URB Entry Allocation Size , 64)))
		Restriction
		The total URB allocation size cannot be greater than 32 KB. This ensures that VFE does not overwrite POSH push constants when both POSH and RCS Compute pipes are active.
		The total URB allocation size for CCS contexts cannot be greater than 32 KB.
		The total URB allocation size cannot be greater than 1024kB.
15:0		CURBE Allocation Size
		Format: U16
		Specifies the total length allocated for CURBE, in 256-bit (32 byte) register increments. Each URB entry is rounded up to be 64-byte aligned. ROB address for CURBE starts at address 64.
		Programming Notes
		CURBE Allocation Size should be 0 for GPGPU and Media workloads that use indirect instead of CURBE. See programming notes and restrictions on the URB Entry Allocation Size for limitations on the total URB size that apply to CURBE Allocation Size.
6	31:0	Reserved
		Format: MBZ
7	31:0	Reserved
		Format: MBZ
8	31:0	Reserved
		Format: MBZ

Media Block Read MSD

MSD1R_MB - Media Block Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: _____ 04h Format: _____ Opcode Media Block Read message
	13:11	Reserved Format: _____ MBZ
	10:8	Vertical Line Stride Override Format: _____ MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	Binding Table Index Format: _____ MDC_BTS Specifies the Binding Table Index for the message

Media Block Write MSD

MSD1W_MB - Media Block Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18:14	Message Type Default Value: 0Ah Format: Opcode Media Block Write message
	13:11	Reserved Format: MBZ
	10:8	Vertical Line Stride Override Format: MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	Binding Table Index Format: MDC_BTS Specifies the Binding Table Index for the message

Memory Fence MSD

MSD_MEMFENCE - Memory Fence MSD								
Source:		EuSubFunctionDataPort0						
Length Bias:		1						
DWord	Bit	Description						
0	31:29	Reserved Format: MBZ						
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.						
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.						
	19	Header Present Format: MDC_MHP Indicates that the message requires a header.						
	18	Legacy Message Default Value: 0h Format: Opcode Legacy Message						
	17:14	Message Type Default Value: 07h Format: Opcode Memory Fence message						
	13	Commit Format: Enable Specifies whether control is returned to the thread only after the fence has been honored. <table border="1" data-bbox="337 1717 1464 1843"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Enabled [Default]</td> <td>The commit writeback register is always required to guarantee ordering.</td> </tr> </tbody> </table>	Value	Name	Description	1	Enabled [Default]	The commit writeback register is always required to guarantee ordering.
	Value	Name	Description					
	1	Enabled [Default]	The commit writeback register is always required to guarantee ordering.					
	12:9	Reserved Format: MBZ						

MSD_MEMFENCE - Memory Fence MSD

8	L1 Flush Data Format: Enable When set, invalidate this subslice's L1 read-only data cache. <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Restriction</div> When "L1 Flush Data" is set, the "L3 Flush" field must be set to 0. If both L1 and L3 needs to be invalidated/flushed, SW need to send two separate fence messages.	
7:0	Binding Table Index Format: MDC_BTS_SLM_A32 Specifies whether global memory or shared local memory (SLM) is fenced with this operation.	
	Value	Name
	Description	
0FEh	SLM	Only SLM is fenced with this operation. Global memory is not fenced. Restriction : The L3 Flush and L1 Flush fields are ignored when SLM memory is selected.
00h	Global Memory [Default]	Only global memory is fenced with this operation. SLM memory is not fenced. Performance : When a program needs to guarantee that all global memory writes are globally observable before a thread retires, a Memory Fence operation is used immediately before the EOT message.

MFC_AVC_PAK_OBJECT

MFC_AVC_PAK_OBJECT				
Source:	VideoCS			
Length Bias:	2			
<p>The MFC_AVC_PAK_OBJECT command is the second primitive command for the AVC Encoding Pipeline. The same command is used for both CABAC and CAVLC modes. The MV Data portion of the bitstream is loaded as indirect data object. Before issuing a MFC_AVC_PAK_OBJECT command, all AVC MFX states need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of this command. MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the Start_MB_In_Slice loaded at the beginning of each slice. MFC_AVC_PAK_OBJECT command follows the MbType definition like MFD. Many fields in this command are identical to that in VME output. This is intended to reduce software converting overhead from VME to PAK. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	28:27	Pipeline	
			Default Value:	2h MFC_AVC_PAK_OBJECT
			Format:	OpCode
	26:24	26:24	Media Command Opcode	
			Default Value:	1h AVC_ENC
			Format:	OpCode
	23:21	23:21	SubOpcode A	
Default Value:			2h	
Format:			OpCode	
20:16	20:16	SubOpcode B		
		Default Value:	9h	
		Format:	OpCode	
15:12	15:12	Reserved		
		Format:	MBZ	
11:0	11:0	DWord Length		
		Default Value:	000Ah DWORD_COUNT_n	
		Format:	=n	
1	31:10	Reserved		
		Format:	MBZ	

MFC_AVC_PAK_OBJECT					
	9:0	<p>Indirect PAK-MV Data Length</p> <p>This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size). Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.</p>			
2	31:29	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
	Format:	MBZ			
28:0	<p>Indirect PAK-MV Data Start Address Offset</p> <p>This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address. Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 60%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[0,512MB)</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,512MB)	
Value	Name				
[0,512MB)					
3..10	255:0	<p>Inline Data</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U32[8]</td> </tr> </table> <p>All the required MB level controls and parameters for encoding are captured as inline data of the MFC_AVC_PAK_OBJECT command. It has a fixed size of 8 DWs. Its definition is described in the next section.</p>	Format:	U32[8]	
		Format:	U32[8]		
<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ				
11	31:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
12..23	383:0	<p>VDEnc Mode Inline Data</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U32[12]</td> </tr> </table> <p>In VDEnc mode, PAK gets inline MVs. These DWs are placed in the PAK Object command in-order to facilitate PAK stand-alone validation mode. Its definition is described in the next section.</p>	Format:	U32[12]	
		Format:	U32[12]		
<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ				

MFC_JPEG_HUFF_TABLE_STATE

MFC_JPEG_HUFF_TABLE_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This Huffman table commands contains both DC and AC tables for either luma or chroma. Once a Huffman table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used in the remaining Scans of the current image. Two Huffman tables for luma and chroma will be sent to H/W, and chroma table is used for both U and V.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFC_JPEG_HUFF_TABLE_STATE
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	7h JPEG
Format:		OpCode	
23:21	SubOpcode A		
	Default Value:	2h Common	
	Format:	OpCode	
20:16	SubOpcode B		
	Default Value:	3h MEDIA_	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0AEh Excludes DWord (0,1)	
	Format:	=n	
1	31:1	Reserved	
		Format:	MBZ
	0	Huff Table ID	
		Format:	U1
		Huffman table destination identifier will specify one of two destinations at the encoder into which the Huffman table must be stored.	
		Value	Name
			Description

MFC_JPEG_HUFF_TABLE_STATE			
		0	Huffman table 0
		1	Huffman table 1
2..13	383:0	DC_TABLE 12 categories with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.	
14..175	5183:0	AC_TABLE 162 run/size with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.	

MFC_JPEG_SCAN_OBJECT

MFC_JPEG_SCAN_OBJECT			
Source:	VideoCS		
Length Bias:	2		
Encoder Only			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFC_JPEG_SCAN_OBJECT
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	7h JPEG_ENC
Format:		OpCode	
23:21	SubOpcode A		
	Default Value:	2h	
	Format:	OpCode	
20:16	SubOpcode B		
	Default Value:	9h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	001h Excludes DWord (0,1)	
	Format:	=n	
1	31:26	Reserved	
		Format:	MBZ
	25:0	MCU Count	
Format:		U26	
<p>This field indicates the number of MCUs in the Scan. MCU Count = $M_x \times M_y$ The number of MCUs in a row: $M_x = (X + (H_1 * 8 - 1)) / (H_1 * 8)$ The number of MCUs in a column: $M_y = (Y + (V_1 * 8 - 1)) / (V_1 * 8)$ X: The number of samples per line in Y-image Y: The number of lines in Y-image H1: Horizontal sampling factor of Y-image in the Frame header V1: Vertical sampling factor of Y-image in the Frame header</p>			
2	31:25	Reserved	

MFC_JPEG_SCAN_OBJECT

	Format:	MBZ																					
24:22	<p>Huffman AC Table</p> <p>AC Huffman table destination selector specifies one of two possible AC table destinations for each Y, U, V, or R, G, B. The AC Huffman tables must have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command.</p> <p>If AC table 0 is used for Y and AC table 1 is used for U and V, it will be set to 110b. If AC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0XXb</td> <td>Bit24 (V0)</td> <td>The third image component must use the AC table 0.</td> </tr> <tr> <td>1XXb</td> <td>Bit24 (V1)</td> <td>The third image component must use the AC table 1.</td> </tr> <tr> <td>X0Xb</td> <td>Bit23 (U0)</td> <td>The second image component must use the AC table 0.</td> </tr> <tr> <td>X1Xb</td> <td>Bit23 (U1)</td> <td>The second image component must use the AC table 1.</td> </tr> <tr> <td>XX0b</td> <td>Bit22 (Y0)</td> <td>The first image component must use the AC table 0.</td> </tr> <tr> <td>XX1b</td> <td>Bit22 (Y1)</td> <td>The first image component must use the AC table 1.</td> </tr> </tbody> </table> <p style="text-align: center; margin-top: 10px;">Restriction</p> <p>When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR, or YUV, Bit22 is used for the first image component, Bit23 is used for the second image component, and Bit24 is used for the third image component.</p>		Value	Name	Description	0XXb	Bit24 (V0)	The third image component must use the AC table 0.	1XXb	Bit24 (V1)	The third image component must use the AC table 1.	X0Xb	Bit23 (U0)	The second image component must use the AC table 0.	X1Xb	Bit23 (U1)	The second image component must use the AC table 1.	XX0b	Bit22 (Y0)	The first image component must use the AC table 0.	XX1b	Bit22 (Y1)	The first image component must use the AC table 1.
Value	Name	Description																					
0XXb	Bit24 (V0)	The third image component must use the AC table 0.																					
1XXb	Bit24 (V1)	The third image component must use the AC table 1.																					
X0Xb	Bit23 (U0)	The second image component must use the AC table 0.																					
X1Xb	Bit23 (U1)	The second image component must use the AC table 1.																					
XX0b	Bit22 (Y0)	The first image component must use the AC table 0.																					
XX1b	Bit22 (Y1)	The first image component must use the AC table 1.																					
21	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <tr> <td style="width: 50%; border-bottom: none;">Format:</td> <td style="width: 50%; border-bottom: none;">MBZ</td> </tr> </table>		Format:	MBZ																			
Format:	MBZ																						
20:18	<p>Huffman DC Table</p> <p>DC Huffman table destination selector specifies one of two possible DC table destinations for each Y, U, V, or R, G, B. The DC Huffman tables shall have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command.</p> <p>if DC table 0 is used for Y and DC table 1 is used for U and V, it will be set to 110b. If DC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0XXb</td> <td>Bit20 (V0)</td> <td>The third image component must use the DC table 0.</td> </tr> <tr> <td>1XXb</td> <td>Bit20 (V1)</td> <td>The third image component must use the DC table 1.</td> </tr> <tr> <td>X0Xb</td> <td>Bit19 (U0)</td> <td>The second image component must use the DC table 0.</td> </tr> <tr> <td>X1Xb</td> <td>Bit19 (U1)</td> <td>The second image component must use the DC table 1.</td> </tr> <tr> <td>XX0b</td> <td>Bit18 (Y0)</td> <td>The first image component must use the DC table 0.</td> </tr> <tr> <td>XX1b</td> <td>Bit18 (Y1)</td> <td>The first image component must use the DC table 1.</td> </tr> </tbody> </table>		Value	Name	Description	0XXb	Bit20 (V0)	The third image component must use the DC table 0.	1XXb	Bit20 (V1)	The third image component must use the DC table 1.	X0Xb	Bit19 (U0)	The second image component must use the DC table 0.	X1Xb	Bit19 (U1)	The second image component must use the DC table 1.	XX0b	Bit18 (Y0)	The first image component must use the DC table 0.	XX1b	Bit18 (Y1)	The first image component must use the DC table 1.
Value	Name	Description																					
0XXb	Bit20 (V0)	The third image component must use the DC table 0.																					
1XXb	Bit20 (V1)	The third image component must use the DC table 1.																					
X0Xb	Bit19 (U0)	The second image component must use the DC table 0.																					
X1Xb	Bit19 (U1)	The second image component must use the DC table 1.																					
XX0b	Bit18 (Y0)	The first image component must use the DC table 0.																					
XX1b	Bit18 (Y1)	The first image component must use the DC table 1.																					

MFC_JPEG_SCAN_OBJECT

Restriction		
When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR, YUV, Bit18 is used for the first image component, Bit19 is used for the second image component, and Bit20 is used for the third image component.		
17	Head Present Flag If this flag is set to 0, then no MFC_JPEG_PAK_INSERT_OBJECT commands will be sent. If this flag is set to 1, then one or more MFC_JPEG_PAK_INSERT_OBJECT commands will be sent after MFC_JPEG_SCAN_OBJECT command.	
	Value	Name
	Description	
0		No insertion into the output bitstream buffer before Scan encoded bitstream
1		Headers, tables, App data insertion into the output bitstream buffer. HW will insert the insertion data before the Scan encoded bitstream.
16	Is Last Scan If this flag is set, then HW will insert EOI (0xFFD9) to the end of Scan encoded bitstream.	
	Value	Name
	Description	
0		Not the last Scan.
1		Indicates that the current Scan is the last one.
15:0	Restart Interval Format: U16 Specifies the number of MCUs in an ECS, except for the last ECS. Restart maker is inserted periodically and it separates the two neighboring ECSs.	
	Value	Name
0-FFFFh		
Programming Notes		
A value of '0' implies that the Scan Data has a single ECS.		

MFC_MPEG2_PAK_OBJECT

MFC_MPEG2_PAK_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>The MFC_MPEG2_PAK_OBJECT command is the second primitive command for the MPEG-2 Encoding Pipeline. Different from AVC, the MV Data portion of the bitstream is loaded as part of MB control data. Before issuing a MFC_MPEG2_PAK_OBJECT command, all MPEG2_MFX states need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of this command. MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the Start_MB_In_Slice loaded at the beginning of each slice.</p> <p>MFC_MPEG2_PAK_OBJECT command follows the MbType definition like MFD. Many fields in this command are identical to that in VME output. This is intended to reduce software converting overhead from VME to PAK. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFC_AVC_PAK_INSERT_OBJECT
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	3h MPEG2
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		2h ENC	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	9h MEDIA_	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0007h Excludes DWord (0,1)	
	Format:	=n	
1..8	255:0	Inline Data	
		Format:	U32[8]

MFC_MPEG2_PAK_OBJECT

		All the required MB level controls and parameters for encoding are captured as inline data of the MFC_MPEG2_PAK_OBJECT command. It has a fixed size of 8 DWs. Its definition is described in the next section
--	--	---

MFC_MPEG2_SLICEGROUP_STATE

MFC_MPEG2_SLICEGROUP_STATE								
Source:	VideoCS							
Length Bias:	2							
<p>This is a slice group level command and can be issued multiple times within a picture that is comprised of multiple slice groups. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).</p>								
DWord	Bit	Description						
0	31:29	Command Type						
		Default Value:	3h PARALLEL_VIDEO_PIPE					
		Format:	OpCode					
	28:27	Pipeline						
		Default Value:	2h MFX_MPEG2_SLICEGROUP_STATE					
		Format:	OpCode					
	26:24	Media Command Opcode						
		Default Value:	3h MPEG2					
		Format:	OpCode					
	23:21	SubOpcode A						
Default Value:		2h MEDIA_						
Format:		OpCode						
20:16	SubOpcode B							
	Default Value:	3h MEDIA_						
	Format:	OpCode						
15:12	Reserved							
	Format:	MBZ						
11:0	DWord Length							
	Default Value:	6h Excludes DWord (0,1)						
	Format:	=n						
1	31	MbRateCtrlFlag- RateControlCounterEnable (Encoder-only)						
		To enable the accumulation of bit allocation for rate control. This field enables hardware Rate Control logic. The rest of the RC control fields are only valid when this field is set to 1. Otherwise, hardware ignores these fields. Note: To reset MB level rate control (QRC), we need to set both bits MbRateCtrlFlag and MbRateCtrlReset to 1 in the new slice						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> </tr> <tr> <td>1h</td> <td>Enable</td> </tr> </tbody> </table>	Value	Name	0h	Disable	1h	Enable
		Value	Name					
0h	Disable							
1h	Enable							
	30	MbRateCtrlReset- ResetRateControlCounter (Encoder-only)						

MFC_MPEG2_SLICEGROUP_STATE

		To reset the bit allocation accumulation counter to 0 to restart the rate control.	
		Value	Name
		0h	Disable
		1h	Enable
29:28	MbRateCtrlMode- RC Trigggle Mode (Encoder-only)		
	Value	Name	Description
	00b		Always Rate Control, whereas RC becomes activeif $\text{sum_act} > \text{sum_target}$ or $\text{sum_act} < \text{sum_target}$
	01b		Gentle Rate Control, whereas RC becomes activeif $\text{sum_act} > \text{upper_midpt}$ or $\text{sum_act} < \text{lower_midpt}$
	10b		Loose Rate Control, whereas RC becomes activeif $\text{sum_act} > \text{sum_max}$ or $\text{sum_act} < \text{sum_min}$
	11b		Reserved
27:24	MbRateCtrlParam- RC Stable Tolerance (Encoder-only)		
	Format:		U4
	This field specifies the tolerance required to deactivate RC once it has been triggered.		
	Value	Name	
	[0, 15]		
23	RateCtrlPanicFlag - RC Panic Enable (Encoder-only)		
	If this field is set to 1, RC enters panic mode when $\text{sum_act} > \text{sum_max}$. RC Panic Type field controls what type of panicbehavior is invoked.		
	Value	Name	
	0	Disable	
	1	Enable	
22	RateCtrlPanicType - RC Panic Type (Encoder-only)		
	This field selects between two RC Panic methods. If it is set to 0, in panic mode, the macroblock QP is maxed out, setting to requested QP + QP_max_pos_mod. If it is set to 1, for an intra macroblock, AC CBPs are set to zero (note that DC CBPs are not modified). For inter macroblocks, AC and DC CBPs are forced to zero.		
	Value	Name	Description
	0h		QP Panic
	1h		CBP Panic
21	Reserved		
	Format:		MBZ
20	SkipConvDisabled - MB Type Skip Conversion Disable (Encoder-only)		
	This field is only valid for a P or B slice. It must be zero for other slice types. Rules are provided in Section 2.3.3.1.6		
	Value	Name	Description
	0h	Enable	Enable skip type conversion

MFC_MPEG2_SLICEGROUP_STATE			
	1h	Disable	Disable skip type conversion
19	IsLastSliceGrp		IsLastSliceGrp = 1 if the current slice group is the last slice group of a picture; 0 otherwise. It is used by the zero filling in the Minimum Frame Size test.
18	BitstreamOutputFlag - Compressed BitStream Output Disable Flag (Encoder-only)		
	Value	Name	Description
	0h	Enable	enable the writing of the output compressed bitstream
	1h	Disable	disable the writing of the output compressed bitstream
17	HeaderPresentFlag - Header Insertion Present in Bitstream (Encoder-only)		
	Value	Name	Description
	0h	Disable	no header insertion into the output bitstream buffer, in front of the current slice encoded bits
	1h	Enable	header insertion into the output bitstream buffer is present, and is in front of the current slice encoded bits.
16	SliceData PresentFlag - SliceData Insertion Present in Bitstream (Encoder-only)		
	Value	Name	Description
	0h	Disable	no Slice Data insertion into the output bitstream buffer
	1h	Enable	Slice Data insertion into the output bitstream buffer is present.
15	TailPresentFlag - Tail Insertion Present in bitstream (Encoder-only)		
	Value	Name	Description
	0h		no tail insertion into the output bitstream buffer, after the current slice encoded bits
	1h		tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.
14	FirstSliceHdrDisabled		
	when this is on, the first slice header of the slice group is expected to be provided by the user via insertion command. PAK HW will skip it.		
13	IntraSlice		
	intra slice value included in slice headers, when IntraSliceFlag = 1.		
12	IntraSliceFlag		
	intra slice flag included in slice headers		
11:8	Reserved		
	Format:	MBZ	
7:4	SliceID[3:0] (Encoder-only)		
	To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP		
3:2	Reserved		
	Format:	MBZ	
1:0	StreamID[1:0] (Encoder-only)		

MFC_MPEG2_SLICEGROUP_STATE						
		To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP				
2	31:24	NextSgMbYcnt - also NextStartVertPos Vertical count of the first MB in the next slice group (Encoder-only)Note: This field restricts total number of MB in the Y direction to 255 or less.				
	23:16	NextSgMbXcnt - also NextStartHorzPos BitFieldDesc				
	15:8	FirstMbYcnt - also CurrStartVertPos Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">U8</td></tr></table> also CurrStartVertPos, Vertical count of the first MB in the current slice group (Encoder-only)		U8		
		U8				
7:0	FirstMbXcnt - also CurrStartHorzPos Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">U8</td></tr></table> Horizontal count of the first MB in the current slice group (Encoder-only)		U8			
	U8					
3	31:9	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ		
		MBZ				
	8	SliceGroupSkip Exists If: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">//Encoder Only</td></tr></table> Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">U1</td></tr></table> All macroblocks are skipped		//Encoder Only		U1
		//Encoder Only				
		U1				
7:6	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ			
	MBZ					
5:0	SliceGroupQp Exists If: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">//Encoder Only</td></tr></table> Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">U6</td></tr></table> Initial slice quality parameter		//Encoder Only		U6	
	//Encoder Only					
	U6					
31:29	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ			
	MBZ					
4	28:0	BitstreamOffset - Indirect PAK-BSE Data Start Address (Write) Exists If: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">//Encoder Only</td></tr></table> This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. This field is only valid for AVC encode mode.		//Encoder Only		
		//Encoder Only				

		MFC_MPEG2_SLICEGROUP_STATE		
		Value	Name	
		[0,512MB)		
5	31:24	MaxQpNegModifier - Magnitude of QP Max Negative Modifier (Encoder-only)		
		Format:	U8	
		This field specifies the lower limit of the QP modifier.		
		Value	Name	
		[0, 51]		
	23:16	MaxQpPosModifier - Magnitude of QP Max Positive Modifier (Encoder-only)		
		Format:	U8	
		This field specifies the upper limit of the QP modifier.		
		Value	Name	
		[0, 51]		
15:12	15:12	ShrinkParam - Shrink Resistance (Encoder-only)		
		Format:	U4	
		This field specifies the additional points added each time decreased correction is invoked.		
		Value	Name	
		[0, 15]		
	11:8	11:8	Shrinkaram - Shrink Init (Encoder-only)	
			Format:	U4
			This field specifies the initial points required to trip decreased control.	
			Value	Name
			[0, 15]	
7:4		7:4	GrowParam - Grow Resistance (Encoder-only)	
			Format:	U4
			This field specifies the additional points added each time increased correction is invoked.	
			Value	Name
			[0, 15]	
	3:0	3:0	GrowParam - Grow Init (Encoder-only)	
			Format:	U4
			This field specifies the initial points required to trip increased control.	
			Value	Name
			[0, 15]	
6		31:24	Reserved	
			Format:	MBZ
		23:20	CorrectPoints - Correct 6 (Encoder-only)	
			Format:	U4
			This field specifies the points used in the lower most RC region when sum_act <= sum_min.	

MFC_MPEG2_SLICEGROUP_STATE									
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0, 15]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0, 15]					
Value	Name								
[0, 15]									
19:16	<p>CorrectPoints - Correct 5 (Encoder-only)</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <td colspan="2">This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.</td> </tr> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0, 15]</td> <td></td> </tr> </tbody> </table>	Format:	U4	This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.		Value	Name	[0, 15]	
Format:	U4								
This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.									
Value	Name								
[0, 15]									
15:12	<p>CorrectPoints - Correct 4 (Encoder-only)</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <td colspan="2">This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.</td> </tr> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0, 15]</td> <td></td> </tr> </tbody> </table>	Format:	U4	This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.		Value	Name	[0, 15]	
Format:	U4								
This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.									
Value	Name								
[0, 15]									
11:8	<p>CorrectPoints - Correct 3 (Encoder-only)</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <td colspan="2">This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt.</td> </tr> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0, 15]</td> <td></td> </tr> </tbody> </table>	Format:	U4	This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt.		Value	Name	[0, 15]	
Format:	U4								
This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt.									
Value	Name								
[0, 15]									
7:4	<p>CorrectPoints - Correct 2 (Encoder-only)</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <td colspan="2">This field specifies the points used in the secondRC region when sum_act > upper_midpt but <= sum_max.</td> </tr> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0, 15]</td> <td></td> </tr> </tbody> </table>	Format:	U4	This field specifies the points used in the secondRC region when sum_act > upper_midpt but <= sum_max.		Value	Name	[0, 15]	
Format:	U4								
This field specifies the points used in the secondRC region when sum_act > upper_midpt but <= sum_max.									
Value	Name								
[0, 15]									
3:0	<p>CorrectPoints - Correct 1 (Encoder-only)</p> <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> <tr> <td colspan="2">This field specifies the points used in the top mostRC region when sum_act > sum_max</td> </tr> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0, 15]</td> <td></td> </tr> </tbody> </table>	Format:	U4	This field specifies the points used in the top mostRC region when sum_act > sum_max		Value	Name	[0, 15]	
Format:	U4								
This field specifies the points used in the top mostRC region when sum_act > sum_max									
Value	Name								
[0, 15]									
7	<p>31:28 CV7 - Clamp Value 7 (Encoder-only)</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> </table>	Exists If:	//Encoder Only						
	Exists If:	//Encoder Only							
	<p>27:24 CV6 - Clamp Value 6 (Encoder-only)</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U4</td> </tr> </table>	Exists If:	//Encoder Only	Format:	U4				
Exists If:	//Encoder Only								
Format:	U4								
<p>23:20 CV5 - Clamp Value 5 (Encoder-only)</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> </table>	Exists If:	//Encoder Only							
Exists If:	//Encoder Only								

MFC_MPEG2_SLICEGROUP_STATE

		Format:	U4																																																																																																																															
19:16	CV4 - Clamp Value 4 (Encoder-only)																																																																																																																																	
	Exists If:	//Encoder Only																																																																																																																																
	Format:	U4																																																																																																																																
15:12	CV3 - Clamp Value 3 (Encoder-only)																																																																																																																																	
	Exists If:	//Encoder Only																																																																																																																																
	Format:	U4																																																																																																																																
11:8	CV2 - Clamp Value 2 (Encoder-only)																																																																																																																																	
	Exists If:	//Encoder Only																																																																																																																																
	Format:	U4																																																																																																																																
7:4	CV1 - Clamp Value 1 (Encoder-only)																																																																																																																																	
	Exists If:	//Encoder Only																																																																																																																																
	Format:	U4																																																																																																																																
3:0	CV0 - Clamp Value 0 (Encoder-only)																																																																																																																																	
	<p>If the magnitude of coefficients at location assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with 2CV0-1. For coefficients at locations marked as 'none', no clamping is performed. The following mappings are only applied to luma and chroma blocks\subblocks containing AC coefficients (blocks\subblocks with only DC coeffs will not be clamped).</p> <p>For 8x8 frame block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr><td>none</td><td>none</td><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td></tr> <tr><td>none</td><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td></tr> <tr><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td></tr> <tr><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td></tr> <tr><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td><td>CV0</td></tr> <tr><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td><td>CV0</td><td>CV0</td></tr> </table> <p>For 8x8 field block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr><td>none</td><td>none</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td></tr> <tr><td>none</td><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV6</td><td>CV5</td><td>CV4</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td><td>CV0</td></tr> <tr><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV0</td></tr> <tr><td>CV5</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td></tr> <tr><td>CV5</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td></tr> </table>			none	none	CV7	CV6	CV5	CV4	CV3	CV3	none	CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0	none	none	CV6	CV5	CV4	CV3	CV2	CV1	none	CV7	CV6	CV5	CV4	CV3	CV2	CV1	CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV1	CV7	CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV6	CV5	CV4	CV4	CV3	CV2	CV1	CV0	CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV0	CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0	CV5	CV5	CV4	CV3	CV2	CV1	CV1
none	none	CV7	CV6	CV5	CV4	CV3	CV3																																																																																																																											
none	CV7	CV6	CV5	CV4	CV3	CV3	CV2																																																																																																																											
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2																																																																																																																											
CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1																																																																																																																											
CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1																																																																																																																											
CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0																																																																																																																											
CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0																																																																																																																											
CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0																																																																																																																											
none	none	CV6	CV5	CV4	CV3	CV2	CV1																																																																																																																											
none	CV7	CV6	CV5	CV4	CV3	CV2	CV1																																																																																																																											
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV1																																																																																																																											
CV7	CV6	CV5	CV4	CV3	CV2	CV2	CV1																																																																																																																											
CV6	CV5	CV4	CV4	CV3	CV2	CV1	CV0																																																																																																																											
CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV0																																																																																																																											
CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0																																																																																																																											
CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0																																																																																																																											

MFD_AVC_BSD_OBJECT

MFD_AVC_BSD_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>The MFD_AVC_BSD_OBJECT command is the only primitive command for the AVC Decoding Pipeline. The same command is used for both CABAC and CAVLD modes. The Slice Data portion of the bitstream is loaded as indirect data object. Before issuing a MFD_AVC_BSD_OBJECT command, all AVC states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_AVC_BSD_OBJECT command.</p>			
Context switch interrupt is not supported by this command.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFD_AVC_BSD_OBJECT
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	1h AVC_DEC
Format:		OpCode	
23:21	SubOpcode A		
	Default Value:	1h	
	Format:	OpCode	
20:16	SubOpcode B		
	Default Value:	8h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	5h Excludes DWord (0,1) = 0005	
	Format:	=n	
1	31:0	Indirect BSD Data Length	
		Format:	U32
<p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC Short Format : It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data</p>			

MFD_AVC_BSD_OBJECT							
		+ Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.					
2	31:29	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ			
		MBZ					
28:0	Indirect BSD Data Start Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">U29</td></tr></table> This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address . Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0. It includes the NAL Header (the NAL Header does not need to perform EMU detection). For AVC Base Layer, it is a single byte. But for MVC, the NAL Header is 4 Bytes long. These NAL Header Unit must be passed to HW in the compressed bitstream buffer. <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="text-align: center; color: blue;">Value</th> <th style="text-align: center; color: blue;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0,512MB)</td> <td></td> </tr> </tbody> </table>		U29	Value	Name	[0,512MB)	
	U29						
Value	Name						
[0,512MB)							
3..5	95:0	Inline Data Format: Inline Data Description for MFD_AVC_BSD_Object All the required Slice Header parameters and error handling settings are captured as InLine Data of the AVC_BSD_OBJECT command. It has a fixed size of 3 DWs. Its definition is described in the following section: Inline Data Description.					
6	31:0	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ			
	MBZ						

MFD_AVC_DPB_STATE

MFD_AVC_DPB_STATE							
Source:	VideoCS						
Length Bias:	2						
<p>This is a frame level state command used only in AVC Short Slice Bitstream Format VLD mode. RefFrameList[16] of interface is replaced with intel Reference Picture Addresses[16] of MFX_PIPE_BUF_ADDR_STATE command. The LongTerm Picture flag indicator of all reference pictures are collected into LongTermPic_Flag[16]. FieldOrderCntList[16][2] and CurrFieldOrderCnt[2] of interface are replaced with intel POCList[34] of MFX_AVC_DIRECTMODE_STATE command.</p>							
DWord	Bit	Description					
0	31:29	Command Type					
		Default Value:	3h PARALLEL_VIDEO_PIPE				
		Format:	OpCode				
	28:27	Pipeline					
		Default Value:	2h MFX_MULTI_DW				
	26:24	Media Command Opcode					
		Default Value:	1h AVC_DEC				
	23:21	SubOpcode A					
		Default Value:	1h				
	20:16	SubOpcode B					
Default Value:		6h					
15:12	Reserved						
	Format:	MBZ					
11:0	DWord Length						
	Default Value:	9h Excludes DWord (0,1)					
	Format:	=n					
1	31:16	LongTermFrame_Flag[16][1 bit]					
		One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame.					
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>the picture is a long term reference picture</td> </tr> <tr> <td>0</td> <td>the picture is a short term reference picture</td> </tr> </tbody> </table>	Value	Name	1	the picture is a long term reference picture	0
	Value	Name					
1	the picture is a long term reference picture						
0	the picture is a short term reference picture						
15:0	Non-ExistingFrame_Flag[16][1 bit]						

MFD_AVC_DPB_STATE																	
		<p>One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>INVALID</td> <td>the reference picture in that entry of RefFrameList[] does not exist anymore.</td> </tr> <tr> <td>0</td> <td>VALID</td> <td>the reference picture in that entry of RefFrameList[] is a valid reference</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the corresponding bit of NonExistingFrameFlags shall be set to 0.</p>	Value	Name	Description	1	INVALID	the reference picture in that entry of RefFrameList[] does not exist anymore.	0	VALID	the reference picture in that entry of RefFrameList[] is a valid reference						
Value	Name	Description															
1	INVALID	the reference picture in that entry of RefFrameList[] does not exist anymore.															
0	VALID	the reference picture in that entry of RefFrameList[] is a valid reference															
2	31:0	<p>UsedForReference_Flag[16][2 bits] One-to-one correspondence with the entries of the Intel RefFrameList[16]. 2 bits per reference frame.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NOT_REFERENCE</td> <td>indicates a frame is "not used for reference".</td> </tr> <tr> <td>1</td> <td>TOP_FIELD</td> <td>bit[0] indicates that the top field of a frame is marked as "used for reference".</td> </tr> <tr> <td>2</td> <td>BOTTOM_FIELD</td> <td>bit[1] indicates that the bottom field of a frame is marked as "used for reference".</td> </tr> <tr> <td>3</td> <td>FRAME</td> <td>bit[1:0] indicates that a frame (or field pair) is marked as "used for reference".</td> </tr> </tbody> </table>	Value	Name	Description	0	NOT_REFERENCE	indicates a frame is "not used for reference".	1	TOP_FIELD	bit[0] indicates that the top field of a frame is marked as "used for reference".	2	BOTTOM_FIELD	bit[1] indicates that the bottom field of a frame is marked as "used for reference".	3	FRAME	bit[1:0] indicates that a frame (or field pair) is marked as "used for reference".
Value	Name	Description															
0	NOT_REFERENCE	indicates a frame is "not used for reference".															
1	TOP_FIELD	bit[0] indicates that the top field of a frame is marked as "used for reference".															
2	BOTTOM_FIELD	bit[1] indicates that the bottom field of a frame is marked as "used for reference".															
3	FRAME	bit[1:0] indicates that a frame (or field pair) is marked as "used for reference".															
3..10	255:0	<p>LTSTFrameNumList[16][16 bits] One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. Depending on the corresponding LongTermFrame_Flag[], the content of this field is interpreted differently.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>LongTermFrame_Flag[i]</td> <td>LTSTFrameNumList[i] represent LongTermFrameIdx.</td> </tr> <tr> <td>0</td> <td>ShortTermFrame_Flag[i]</td> <td>LTSTFrameNumList[i] represent Short Term Picture FrameNum.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the LTSTFrameNumList entry shall be set to 0.</p>	Value	Name	Description	1	LongTermFrame_Flag[i]	LTSTFrameNumList[i] represent LongTermFrameIdx.	0	ShortTermFrame_Flag[i]	LTSTFrameNumList[i] represent Short Term Picture FrameNum.						
Value	Name	Description															
1	LongTermFrame_Flag[i]	LTSTFrameNumList[i] represent LongTermFrameIdx.															
0	ShortTermFrame_Flag[i]	LTSTFrameNumList[i] represent Short Term Picture FrameNum.															
11..18	255:0	<p>ViewIDList[16][16 bits] One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. The view ids are 10-bits, the upper 6 bits are ignored. "000000" & ViewId1[9:0] & "000000" & ViewId0[9:0]</p> <p style="text-align: center;">Programming Notes</p> <p>When an Intel RefFrameList[i] is not an valid entries, Viewid should be set to 0x00</p>															
19..22	127:0	<p>ViewOrderListL0[16][8 bits] One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference</p>															

MFD_AVC_DPB_STATE	
	<p>frame. The view order need 4-bits, the upper 4 bits are ignored.0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0]</p> <p style="text-align: center;">Programming Notes</p> <p>When the ViewOrderListL0[i] is not an valid inter-view reference, its corresponding ViewOrder should be set to 0xF</p> <p>Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.</p>
23..26	<p>127:0 ViewOrderListL1[16][8 bits]</p> <p>One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference frame. The view order need 4-bits, the upper 4 bits are ignored.0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0]</p> <p style="text-align: center;">Programming Notes</p> <p>When the ViewOrderListL1[i] is not an valid inter-view reference, its corresponding ViewOrder should be set to 0xF</p> <p>Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.</p>

MFD_AVC_PICID_STATE

MFD_AVC_PICID_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This is a frame level state command used for both AVC Long and Short Format in VLD mode. PictureID[16] contains the pictureID of each reference picture (16 maximum) so hardware can uniquely identify the reference picture across frames (this will be used for DMV operation). This command will be needed for both short and long format.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_MULTI_DW
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	1h MFD_AVC_DPB_STATE
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		1h DEC	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	5h MEDIA_	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0008h Excludes DWord (0,1)	
	Format:	=n	
1	31:1	Reserved	
		Format:	MBZ
0	PictureID Remapping Disable		
	Value	Name	Description
	0h	AVC decoder will use 16 bits Picture ID to handle DMV and identify the reference picture	Desc
	1h	AVC decoder will use 4 bits FrameStoreID (index to RefFrameList) to handle DMV and identify the reference picture (This causes DMV logic to	Desc

MFD_AVC_PICID_STATE		
		function the same.)
		Programming Notes
		If Picture ID Remapping Disable is "1", PictureIDList will not be used.
2..9	255:0	<p>PictureIDList[16][16 bits]</p> <p>One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. PictureID of each Frame uniquely identifies the reference picture across frames. The same number cannot be reused until the reference picture is completely retired(no longer used for reference)When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the LTSTFrameNumList entry shall be set to 0.</p>

MFD_AVC_SLICEADDR

MFD_AVC_SLICEADDR			
Source:	VideoCS		
Length Bias:	2		
<p>This is a Slice level command used only for AVC Short Slice Bitstream Format VLD mode. When decoding a slice, H/W needs to know the last MB of the slice has reached in order to start decoding the next slice. It also needs to know if a slice is terminated but the last MB has not reached, error concealment should be invoked to generate those missing MBs. For AVC Short Format, the only way to know the last MB position of the current slice, H/W needs to snoop into the next slice's start MB address (a linear address encoded in the Slice Header). Since each BSD Object command can have only one indirect bitstream buffer address, this command is added to help H/W to snoop into the next slice's slice header and retrieve its Start MB Address. This command will take the next slice's bitstream buffer address as input (exactly the same way as a BSD Object command), and parse only the first_mb_in_slice syntax element. The result will stored inside the H/W, and will be used to decode the current slice specified in the BSD Object command. Only the very first few bytes (max 5 bytes for a max 4K picture) of the Slice Header will be decoded, the rest of the bitstream are don't care. This is because the first_mb_in_slice is encoded in Exponential Golomb, and will take 33 bits to represent the max $256 \times 256 = 64K-1$ value. The indirect data of MFD_AVC_SLICEADDR is a valid BSD object and is decoded as in BSD OBJECT command. The next Slice Start MB Address is also exposed to the MMIO interface. The Slice Start MB Address (first_mb_in_slice) is a linear MB address count; but it is translated into the corresponding 2D MB X and Y raster position, and are stored internally as NextSliceMbY and NextSliceMbX.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFD_AVC_SLICEADDR
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	1h AVC_DEC
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		1h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	7h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		

MFD_AVC_SLICEADDR													
		<table border="1"> <tr> <td>Default Value:</td> <td>2h</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	2h	Format:	=n							
Default Value:	2h												
Format:	=n												
1	31:0	<p>Indirect BSD Data Length</p> <table border="1"> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. Driver always programs this up to 5 bytes; for bitstream less than 5 bytes, driver program the lesser value. (Emulation Prevention Byte should never happen for the first 5 bytes when the max picture size can only be 4Kx4K)It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.</p>	Format:	U32									
Format:	U32												
2	31:29	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	28:0	<p>Indirect BSD Data Start Address</p> <p>This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0.It includes the NAL Header Byte. (but does not perform EMU detection).Must provide a valid MB address, even if error. MB must be clamped to within a pic boundary.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,512MB)</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,512MB)								
Value	Name												
[0,512MB)													
3	31:13	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	12:9	<p>Reserved</p>											
	8	<p>AVC NAL Type First Byte Override Bit</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>This bit indicates hardware should use the NAL Type (provided below) programmed by driver instead of using the one from bitstream. The NAL byte from bitstream will not be correct.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Use Bitstream Decoded NAL Type</td> <td>NAL Type should come from first byte of decoded bitstream.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Use Driver Programmed NAL Type</td> <td>NAL Type should come from "Driver Provided NAL Type Values" programmed by driver.</td> </tr> </tbody> </table>	Format:	U1	Value	Name	Description	0	Use Bitstream Decoded NAL Type	NAL Type should come from first byte of decoded bitstream.	1	Use Driver Programmed NAL Type	NAL Type should come from "Driver Provided NAL Type Values" programmed by driver.
Format:	U1												
Value	Name	Description											
0	Use Bitstream Decoded NAL Type	NAL Type should come from first byte of decoded bitstream.											
1	Use Driver Programmed NAL Type	NAL Type should come from "Driver Provided NAL Type Values" programmed by driver.											
	7:0	<p>Driver Provided NAL Type Value</p> <table border="1"> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>This will replace the first byte of the NAL unit, containing forbidden_zero_bit, nal_ref_idc, and nal_unit_type.</p>	Format:	U8									
Format:	U8												

MFD_AVC_SLICEADDR		
		Programming Notes
		This byte should be ignored if AVC NAL Type First Byte Override Bit is programmed to 0

MFD_IT_OBJECT

MFD_IT_OBJECT				
Source:	VideoCS			
Length Bias:	2			
All weight mode (default and implicit) are mapped to explicit mode. But the weights come in either as explicit or implicit.				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	Pipeline		
		Default Value:	2h MFD_IT_OBJECT	
	26:24	Media Command Opcode		
		Default Value:	0h MFX_COMMON_DEC	
	23:21	SubOpcode A		
		Default Value:	1h	
	20:16	SubOpcode B		
Default Value:		9h		
15:12	Reserved			
	Format:	MBZ		
11:0	DWord Length			
	Format:	=n		
	Value	Name	Description	Exists If
	0Ch	AVC	There are total of 7 inline DWs for AVC (in addition to DW0-DW6 here)	//mode=='AVC'
	10h	VC1	There are total of 11 inline DWs for VC1 (in addition to DW0-DW6 here)	//mode=='VC1'
0Bh	MPEG2	There are total of 6 inline DWs for AVC (in addition to DW0-DW6 here)	//mode=='MPEG2'	
1	31:10	Reserved		
		Format:	MBZ	
	9:0	Indirect IT-MV Data Length		

MFD_IT_OBJECT					
		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U10</td> </tr> </table> <p>This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC-IT Mode: It must be DWord aligned (since each MV is 4bytes in size)Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV. This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data).</p>	Format:	U10	
Format:	U10				
2	31:29	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
	Format:	MBZ			
28:0	<p>Indirect IT-MV Data Start Address Offset</p> <p>This field specifies the memory starting address (offset) of the MV data to be fetched into the IT pipeline for processing. This pointer is relative to the Indirect IT-MV Object Base Address. Hardware ignores this field if indirect data is not present, i.e. the Indirect MV Data Length is set to 0. Alignment of this address depends on the mode of operation. AVC-IT Mode: It must be DWord aligned (since each MV is 4 bytes in size). This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data).</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 60%; text-align: center;">Value</th> <th style="width: 40%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0,512MB)</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,512MB)	
Value	Name				
[0,512MB)					
3	31:12	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
	Format:	MBZ			
11:0	<p>Indirect IT-COEFF Data Length</p> <p>This field provides the length in bytes of the indirect data, which contains all the non-zero coefficients for the current MB. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-COEFF Data Start Address field is ignored. Since each IT-COEFF data is 1 DW in size, with 12 bits, this field can be extended to support up to 4:4:4 format.(256 pixel * 3 byte pixel components * 4 bytes per coeff).This field must be integer multiple of 16-bytes for AVC (since each coefficient is 4 bytes in size).This field is only valid in AVC, VC1, MPEG2 decoder IT mode.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 45%; text-align: center;">Value</th> <th style="width: 55%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0,3072]</td> <td style="text-align: center;">In bytes [0, 256*3*4]</td> </tr> </tbody> </table>	Value	Name	[0,3072]	In bytes [0, 256*3*4]
Value	Name				
[0,3072]	In bytes [0, 256*3*4]				
4	31:29	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
	Format:	MBZ			
28:0	<p>Indirect IT-COEFF Data Start Address Offset</p> <p>This field specifies the memory starting address (offset) of the coeff data to be loaded into the IT pipeline for processing. This pointer is relative to the Indirect IT-COEFF Object Base Address. Hardware ignores this field if indirect IT-COEFF data is not present, i.e. the Indirect IT-COEFF Data Length is set to 0.This field must be DW aligned, since each coefficient is 4 bytes in size. Driver will determine the Num of EOB 4x4/8x8 must match the block cbp flags, if not match, hardware cannot hang - add</p>				

MFD_IT_OBJECT							
		<p>error handling. This field is only valid in AVC, VC1, MPEG2 decoder IT mode.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 60%;">Value</th> <th style="width: 40%;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,512MB)</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,512MB)		
Value	Name						
[0,512MB)							
5	31:6	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ			
	Format:	MBZ					
5:0	<p>Indirect IT-DBLK Control Data Length</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U6</td> </tr> </table> <p>This field provides the length in bytes of the indirect data, which contains all the deblocker control information for the current MB (in 4x4 sub-block partitioning). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-DBLK Data Start Address field is ignored. This field must have the same alignment as the Indirect IT-DBLK Data Start Address. It must be DWord aligned. Each Deblock Control Data record is 48 bytes or 12 DWords in size. This field is only valid in AVC decoder IT mode.</p>	Format:	U6				
Format:	U6						
6	31:29	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ			
	Format:	MBZ					
28:0	<p>Indirect IT-DBLK Control Data Start Address Offset</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">IndirectObjectOffset[28:0]</td> </tr> </table> <p>This field specifies the memory starting address (offset) of the Deblocker control data to be fetched into the IT Pipeline for processing. This pointer is relative to the Indirect IT-DBLK Object Base Address. Hardware ignores this field if indirect data is not present, ie. The indirect IT-DBLK Control Data Length is set to 0. It must be DWord aligned. Each Deblock Control Data record is 48 bytes or 12 DWords in size. This field is only valid in AVC decoder IT mode.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 60%;">Value</th> <th style="width: 40%;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,512MB)</td> <td></td> </tr> </tbody> </table>	Format:	IndirectObjectOffset[28:0]	Value	Name	[0,512MB)	
Format:	IndirectObjectOffset[28:0]						
Value	Name						
[0,512MB)							
7..12 Exists if: //mode == 'MPEG2' Programming Notes: MPEG2--There are 6 addition DWs so n = 12	191:0	<p>MPEG2 Inline Data</p> <p>Union for all 3 codecs. Includes IT, MC, IntraPred inline data as well as Deblocker control information.</p> <p>AVC-IT Modes: Hardware interprets this data in the specified format.</p> <p>VC1-IT Modes: Hardware interprets this data in the specified format. MV inline.</p> <p>MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inlineFor AVC there 7 DWords of inline data, hence N is equal to 13.</p>					
7..13 Exists if: //mode == 'AVC' Programming Notes: AVC--There are 7 addition DWs so	223:0	<p>AVC Inline Data</p> <p>Union for all 3 codecs. Includes IT, MC, IntraPred inline data as well as Deblocker control information.</p> <p>AVC-IT Modes: Hardware interprets this data in the specified format.</p> <p>VC1-IT Modes: Hardware interprets this data in the specified format. MV inline</p> <p>MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inline. For AVC there 7 DWords of inline data, hence N is equal to 13.</p>					

MFD_IT_OBJECT		
n = 13		
7..17 Exists if: //mode == 'VC1' Programming Notes: VC1--There are 11 addition DWs so n = 17	351:0	VC1 Inline Data Union for all 3 codecs. Includes IT, MC, IntraPred inline data as well as Deblocker control information. AVC-IT Modes: Hardware interprets this data in the specified format. VC1-IT Modes: Hardware interprets this data in the specified format. MV inline. MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inline. For AVC there 7 DWords of inline data, hence N is equal to 13.

MFD_JPEG_BSD_OBJECT

MFD_JPEG_BSD_OBJECT		
Source:	VideoCS	
Length Bias:	2	
Exists If:	//Decoder	
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h PARALLEL_VIDEO_PIPE
		Format: OpCode
	28:27	Pipeline
		Default Value: 2h MFD_JPEG_BSD_OBJECT
	26:24	Media Command Opcode
		Default Value: 7h JPEG_DEC
	23:21	SubOpcode A
Default Value: 1h		
20:16	SubOpcode B	
	Default Value: 8h	
15:12	Reserved	
	Format: MBZ	
11:0	DWord Length	
	Default Value: 004h Excludes DWord (0,1)	
1	31:0	Indirect Data Length
		. It is the length in bytes of the bitstream data for the current Scan. It includes the first byte of the first MCU and the last non-zero byte of the last MCU in the Scan. Specifically, the zero-padding bytes (if present) are excluded. Hardware ignores the contents after the last non-zero byte.
2	31:29	Reserved
		Format: MBZ
	28:0	Indirect Data Start Address
		Format: IndirectObjectOffset[28:0]
This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address. Hardware ignores		

MFD_JPEG_BSD_OBJECT											
		this field if indirect data is not present. It is a byte-aligned address for the JPEG bitstream data									
3	31:29	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">MBZ</td></tr></table>		MBZ							
		MBZ									
	28:16	Scan Horizontal Position Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">U13</td></tr></table> This field indicates the horizontal position (in block units) of the first MCU in the Scan.		U13							
		U13									
15:13	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">MBZ</td></tr></table>		MBZ								
	MBZ										
12:0	Scan Vertical Position Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">U13</td></tr></table> This field indicates the vertical position (in block units) of the first MCU in the Scan.		U13								
	U13										
4	31	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">MBZ</td></tr></table>		MBZ							
		MBZ									
	30	Interleaved <table border="1" style="width: 100%; text-align: center;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Non-Interleaved</td> <td>one component in the Scan</td> </tr> <tr> <td>1</td> <td>Interleaved</td> <td>multiple components in the Scan</td> </tr> </tbody> </table>	Value	Name	Description	0	Non-Interleaved	one component in the Scan	1	Interleaved	multiple components in the Scan
	Value	Name	Description								
	0	Non-Interleaved	one component in the Scan								
1	Interleaved	multiple components in the Scan									
29:27	Scan Components Bit0: YBit1: UBit2: V For example, if non-interleaved Y, then it will be set to 001b. If interleaved Y, U, and V, it will be set to 111b.										
26	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">MBZ</td></tr></table>		MBZ								
	MBZ										
25:0	MCU Count Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">U26</td></tr></table> This field indicates the number of MCUs in the Scan.		U26								
	U26										
5	31:16	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">MBZ</td></tr></table>		MBZ							
		MBZ									
15:0	RestartInterval(16 bit) Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%;">U16</td></tr></table> Specifies the number of MCU in restart interval. Valid values are 1->0xFFFFValue of 0 implies that all the SCAN have only one ECS.		U16								
	U16										

MFD_MPEG2_BSD_OBJECT

MFD_MPEG2_BSD_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>Different from AVC and VC1, MFD_MPEG2_BSD_OBJECT command is able to be pipelined. This is for performance purpose as in MPEG2 a slice is defined as a group of MBs of any size that must be within a macroblock row. Slice header parameters are passed in as inline data and the bitstream data for the slice is passed in as indirect data. Of the inline data, slice_horizontal_position and slice_vertical_position determines the location within the destination picture of the first macroblock in the slice. The content in this command is identical to that in the MEDIA_OBJECT command in VLD mode described in the Media Chapter.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFD_MPEG2_BSD_OBJECT
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	3h MPEG2_DEC
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		1h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	8h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0003h Excludes DWord (0,1)	
	Format:	=n	
1	31:0	Indirect BSD Data Length	
		Format:	U32
<p>It is the length in bytes of the bitstream data for the current slice. It includes the first byte of the first macroblock and the last non-zero byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. This field is sized to support beyond MPEG-2 MP@HL bitstream (<4K). According to Table 8-6 of ISO/IEC 13818-2, the maximum number of bits per macroblock for 4:2:0 is 4608. So the maximum slice size for 4K</p>			

MFD_MPEG2_BSD_OBJECT							
	<p>x 4K is $4608 * 256 / 8 = 147,456$ bytes (0x24000), which requires 18 bits.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">As MPEG-2 spec does not post any limitation of the size of zero-padding bytes, it is possible to have a slice data with large length (including zero-padding bytes). As the data beyond 0x10E00 would only be zero bytes for a valid slice data</td> </tr> <tr> <td colspan="2">zero-padding restriction is removed</td> </tr> </table>	Programming Notes		As MPEG-2 spec does not post any limitation of the size of zero-padding bytes, it is possible to have a slice data with large length (including zero-padding bytes). As the data beyond 0x10E00 would only be zero bytes for a valid slice data		zero-padding restriction is removed	
Programming Notes							
As MPEG-2 spec does not post any limitation of the size of zero-padding bytes, it is possible to have a slice data with large length (including zero-padding bytes). As the data beyond 0x10E00 would only be zero bytes for a valid slice data							
zero-padding restriction is removed							
2	31:29	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
	Format:	MBZ					
28:0	<p>Indirect Data Start Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>IndirectObjectOffset[28:0]</td> </tr> </table> <p>This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the MPEG2 VLD bitstream data This address points to the first byte of the MB layer data, i.e. not including slice header.</p>	Format:	IndirectObjectOffset[28:0]				
Format:	IndirectObjectOffset[28:0]						
3..4	63:0	<p>Inline Data</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MFD_MPEG2_BSD_OBJECT Inline Data Description</td> </tr> </table> <p>All the required Slice Header parameters and error handling settings are captured as MPEG2_BSD_OBJECT Inline Data Descriptor structures. It has a fixed size of 2 DWs. Its definition is described in the next section.</p>	Format:	MFD_MPEG2_BSD_OBJECT Inline Data Description			
Format:	MFD_MPEG2_BSD_OBJECT Inline Data Description						

MFD_VC1_BSD_OBJECT

MFD_VC1_BSD_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>The MFD_VC1_BSD_OBJECT command is the only primitive command for the VC1 Decoding Pipeline. The macroblock data portion of the bitstream is loaded as indirect data object. Before issuing a MFD_VC1_BSD_OBJECT command, all VC1 states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_VC1_BSD_OBJECT command. VC1 deblock filter kernel cross the slice boundary if in the last MB row of a slice, so need to know the last MB row of a slice to disable the edge mask. There is why VC1 BSD hardware need to know the end of MB address for the current slice. As such no more phantom slice is needed for VC1, as long as the driver will program both start MB address in the current slice and the start MB address of the next slice. As a result, we can also support multiple picture state commands in between slices.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_MULTI_DW
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	2h VC1_DEC
Format:		OpCode	
23:21	SubOpcode A		
	Default Value:	1h	
	Format:	OpCode	
20:16	SubOpcode B		
	Default Value:	8h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0003h Excludes DWord (0,1)	
	Format:	=n	
1	31:24	Reserved	
		Format:	MBZ
	23:0	Indirect BSD Data Length	

MFD_VC1_BSD_OBJECT

		Format:	U24
		<p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. Long Format : It is the length in bytes of the bitstream data for the current slice/picture. It includes the first byte of the first macroblock and the last byte of the last macroblock in the slice/picture. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. Hardware ignores the contents after the last non-zero byte (trailing zeros). This field is sized to support VC1 AP@L4 Level bitstream. It includes the byte that contains the First MB Bit Offset Short Format : It is the length in bytes of the bitstream data for the current slice, including Picture/Slice Header + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly.</p>	
2	31:29	Reserved	
		Format:	MBZ
	28:0	Indirect Data Start Address	
		Format:	IndirectObjectOffset[28:0]
		<p>This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VC1 bitstream data.</p>	
		Value	Name
		[0,512MB)	
3	31:24	Reserved	
		Format:	MBZ
	23:16	Slice Start Vertical Position	
		<p>This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. For SecondField this value is reset to zero as opposed to the VC1 spec Ref: 9.1.2 Slice Layer. This field is for both Long and Short VC1 Interface Format.</p>	
	15:9	Reserved	
		Format:	MBZ
	8:0	Next Slice Vertical Position	
		<p>This field specifies the position in y-direction of the first macroblock in the next Slice in unit of macroblocks. This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to the height of picture (since y-direction is zero-based numbering)This field is maintained and provided by the driver for both Long and Short VC1 Interface Format.</p>	
4	31:16	First_MB_Byte_Offset_of_Slice_Data_or_Slice_Header	
		<p>For DXVA2 VC1 Short Format only. It gives the byte offset to locate the first MB data in the bitstream for a slice, relative to the Indirect BSD Data Start Address.</p>	
	15:5	Reserved	
		Format:	MBZ

MFD_VC1_BSD_OBJECT				
	4	Emulation Prevention Byte Present		
		Value	Name	Description
		0h		H/W needs to perform Emulation Byte Removal
	1h		H/W does not need to perform Emulation Byte Removal	
	3	Reserved		
		Format:		MBZ
	2:0	FirstMbBitOffset (First Macroblock Bit Offset)		
		Format:		U3
		This field provides the bit offset of the first macroblock of the Slice in the first byte of the input compressed bitstream. It is used with First_MB_Byte_Offset for non-byte aligned position.		

MFD_VC1_LONG_PIC_STATE

MFD_VC1_LONG_PIC_STATE	
Source:	VideoCS
Length Bias:	2
<p>MFX_VC1_LONG_PIC_STATE command encapsulates the decoding parameters that are read or derived from bitstream syntax elements above (inclusive) picture header layer. These parameters are static for a picture and when slice structure is present, these parameters are not changed from slice to slice of the same picture. Hence, this command is only issued at the beginning of processing a new picture and prior to the VC1*_OBJECT command. The values set for these state variables are retained internally across slices. Only the parameters needed by hardware (BSD unit) to decode bit sequence for the macroblocks in a picture layer or a slice layer are presented in this command. Other parameters such as the ones used for inverse transform or motion compensation are provided in MFX_VC1_PRED_PIPE_STATE command. This Long interface format is intel proprietary interface. Driver will need to perform addition operations to generate all the fields in this command.</p>	

DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFD_VC1_LONG_PIC_STATE
	26:24	Media Command Opcode	
		Default Value:	2h VC1_DEC
	23:21	SubOpcode A	
Default Value:		1h	
20:16	SubOpcode B		
	Default Value:	1h	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0004h Excludes DWord (0,1)	
	Format:	=n	
1	31:24	Reserved	
		Format:	MBZ

MFD_VC1_LONG_PIC_STATE

	23:16	PictureHeightInMBsMinus1 (Picture Height Minus 1 in Macroblocks) <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U8</td> </tr> </table> <p>This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead). This field is used in VLD and IT modes.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 30%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,255]</td> <td>ValueHeight</td> <td>Represents 1 MB to 256 MB</td> </tr> </tbody> </table> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="text-align: left;"> Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boundary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension. </td> </tr> </tbody> </table>	Format:	U8	Value	Name	Description	[0,255]	ValueHeight	Represents 1 MB to 256 MB	Programming Notes	Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boundary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.
	Format:	U8										
	Value	Name	Description									
[0,255]	ValueHeight	Represents 1 MB to 256 MB										
Programming Notes												
Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boundary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.												
15:8	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
7:0	PictureWidthInMBsMinus1 (Picture Width Minus 1 in Macroblocks) <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U8-1</td> </tr> </table> <p>This field indicates the width of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureWidthInMBs equals 120 (1920 divided by 16). This field is used in VLD and IT modes</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 30%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,255]</td> <td>ValidWidth</td> <td>Represents 1 MB to 256 MB</td> </tr> </tbody> </table>	Format:	U8-1	Value	Name	Description	[0,255]	ValidWidth	Represents 1 MB to 256 MB			
Format:	U8-1											
Value	Name	Description										
[0,255]	ValidWidth	Represents 1 MB to 256 MB										
2	31:24	Bitplane Buffer Pitch Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U8-1</td> </tr> </table> <p>Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format, it is written by an application and later read by the HW. But in VC1 Short Format, it is written and read by H/W only. This field is specified for better performance</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 60%;">Value</th> <th style="width: 40%;">Name</th> </tr> </thead> <tbody> <tr> <td>[0h, FFh]</td> <td></td> </tr> </tbody> </table> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="text-align: left;"> The pitch must be equal to PictureWidthInMBs/2. For VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2. For VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row. This field is not used in IT mode, used in VLD mode only. For VC1 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver. </td> </tr> </tbody> </table>	Format:	U8-1	Value	Name	[0h, FFh]		Programming Notes	The pitch must be equal to PictureWidthInMBs/2. For VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2. For VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row. This field is not used in IT mode, used in VLD mode only. For VC1 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.		
	Format:	U8-1										
Value	Name											
[0h, FFh]												
Programming Notes												
The pitch must be equal to PictureWidthInMBs/2. For VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2. For VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row. This field is not used in IT mode, used in VLD mode only. For VC1 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.												
23:16	Reserved											

MFD_VC1_LONG_PIC_STATE

	Format:	MBZ															
15	<p>DmvSurfaceValid Indicated when the DMV read surface is valid. This surface stored the direct motion vectors and Mb type. This field is set for B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture). When the current picture being decoded is an I, P or BI, this bit is set to 0, since there is no DMV read in these picture decoding process. This field is not used in IT mode, used in VLD mode only.</p>																
14	<p>ImplicitQuantizer Derived by driver from QUANTIZER. This field is used in intel VC1 VLD Long Format only, not used in IT and VC1. This bit is set to 1 when syntax element QUANTIZER=0, else its set to 0</p>																
13	<p>Interpolation Rounder Contro Used only in MC operation. This field specifies the rounding control value used in interpolation operation of motion prediction process. This field is used in VLD and IT modes.</p>																
Programming Notes																	
This bit field is taken from bRcontrol in PictureParameters data structure																	
12	<p>SyncMarker Indicates whether sync markers are enabled/disabled. If enable, sync markers "may be" present in the current video sequence being decoded. It is a sequence level syntax element and is valid only for Simple and Main Profiles.</p>																
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 25%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Not Present</td> <td>Sync Marker is not present in the bitstream</td> </tr> <tr> <td>1h</td> <td>Maybe present</td> <td>Sync Marker maybe present in the bitstream</td> </tr> </tbody> </table>			Value	Name	Description	0h	Not Present	Sync Marker is not present in the bitstream	1h	Maybe present	Sync Marker maybe present in the bitstream						
Value	Name	Description															
0h	Not Present	Sync Marker is not present in the bitstream															
1h	Maybe present	Sync Marker maybe present in the bitstream															
Programming Notes																	
This field is only valid in VLD mode. For Simple Profile, SyncMarker must set to 0. For Main Profile, SyncMarker can be set to 0 or 1. This field is used in both intel and MS VLD interface, but not used in IT mode.																	
11:8	<p>Motion Vector Mode This field indicates one of the following motion compensation interpolation modes for P and B pictures. The MC interpolation modes apply to prediction values of luminance blocks and are always in quarter-sample. For chrominance blocks, it always performs bilinear interpolation with either half-pel or quarter-pel precision. Before the polarity of Chroma Half-pel or Q-pel is reversed from Spec, now I have fixed it to match with VC1 Spec.</p>																
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0XX0b</td> <td></td> <td>Chroma Quarter -pel + Luma bicubic. (can only be 1MV)</td> </tr> <tr> <td>0XX1b</td> <td></td> <td>Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)</td> </tr> <tr> <td>1XX0b</td> <td></td> <td>Chroma Quarter -pel + Luma bilinear. (can only be 1MV)</td> </tr> <tr> <td>1XX1b</td> <td></td> <td>Chroma Half-pel + Luma bilinear</td> </tr> </tbody> </table>			Value	Name	Description	0XX0b		Chroma Quarter -pel + Luma bicubic. (can only be 1MV)	0XX1b		Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)	1XX0b		Chroma Quarter -pel + Luma bilinear. (can only be 1MV)	1XX1b		Chroma Half-pel + Luma bilinear
Value	Name	Description															
0XX0b		Chroma Quarter -pel + Luma bicubic. (can only be 1MV)															
0XX1b		Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)															
1XX0b		Chroma Quarter -pel + Luma bilinear. (can only be 1MV)															
1XX1b		Chroma Half-pel + Luma bilinear															

MFD_VC1_LONG_PIC_STATE

Programming Notes											
<p>Bits 11:8 are taken from bMVprecisionAndChromaRelation in PictureParameters data structure. Bit 11 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MC. Bit 8 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion. This field is used in both VLD and IT modes.</p>											
7	<p>RangeReductionScale This field specifies whether the reference picture pixel values should be scaled up or scaled down on-the-fly, if RangeReduction is Enabled.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td></td> <td>Scale down reference picture by factor of 2</td> </tr> <tr> <td style="text-align: center;">1h</td> <td></td> <td>Scale up reference picture by factor of 2</td> </tr> </tbody> </table>		Value	Name	Description	0h		Scale down reference picture by factor of 2	1h		Scale up reference picture by factor of 2
Value	Name	Description									
0h		Scale down reference picture by factor of 2									
1h		Scale up reference picture by factor of 2									
<p style="text-align: center;">Programming Notes</p> <p>This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture only. Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.</p>											
6	<p>RangeReduction Enable This field specifies whether on-the-fly pixel value range reduction should be performed for the preceding (or forward) reference picture. Along with RangeReductionScale to specify whether scale up or down should be performed. It is not the same value as RANGEREDFRM Syntax Element (PictureParameters bPicDeblocked bit 5) in the Picture Header.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">Disable</td> <td>Range reduction is not performed</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">Enable</td> <td>Range reduction is performed</td> </tr> </tbody> </table>		Value	Name	Description	0h	Disable	Range reduction is not performed	1h	Enable	Range reduction is performed
Value	Name	Description									
0h	Disable	Range reduction is not performed									
1h	Enable	Range reduction is performed									
<p style="text-align: center;">Programming Notes</p> <p>This field is for Main Profile only. Simple Profile is always disable, and not applicable to Advanced Profile. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture only. Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.</p>											
5	<p>Reserved</p>										
4	<p>Overlap Smoothing Enable Flag This field is the decoded syntax element OVERLAP in bitstream. Indicates if Overlap smoothing is</p>										

MFD_VC1_LONG_PIC_STATE																	
		<p>ON at the picture level This field is used in both VLD and IT modes.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>to disable overlap smoothing filter</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>to enable overlap smoothing filter</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	to disable overlap smoothing filter	1h	Enable	to enable overlap smoothing filter						
Value	Name	Description															
0h	Disable	to disable overlap smoothing filter															
1h	Enable	to enable overlap smoothing filter															
	3	<p>Secondfield This flag is set for the second field in field pictures. This field is used in both VLD and IT modes.</p>															
	2:1	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	0	<p>VC1 Profile specifies the bitstream profile. This field is used in both VLD and IT modes.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>current picture is in Advanced Profile</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not.</p>	Value	Name	Description	0h	Disable	current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)	1h	Enable	current picture is in Advanced Profile						
Value	Name	Description															
0h	Disable	current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)															
1h	Enable	current picture is in Advanced Profile															
3	31	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	30:29	<p>CondOver This field is the decoded syntax element CONDOVER in a bitstream of advanced profile. It controls the overlap smoothing filter operation for an I frame or an BI frame when the picture level qualization step size PQUANT is 8 or lower. This field is used in intel VC1 VLD mode only, not in VC1 and IT modes.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td></td> <td>No overlap smoothing</td> </tr> <tr> <td>01b</td> <td></td> <td>Reserved</td> </tr> <tr> <td>10b</td> <td></td> <td>Always perform overlap smoothing filter</td> </tr> <tr> <td>11b</td> <td></td> <td>Overlap smoothing on a per macroblock basis based on OVERFLAGS</td> </tr> </tbody> </table>	Value	Name	Description	00b		No overlap smoothing	01b		Reserved	10b		Always perform overlap smoothing filter	11b		Overlap smoothing on a per macroblock basis based on OVERFLAGS
Value	Name	Description															
00b		No overlap smoothing															
01b		Reserved															
10b		Always perform overlap smoothing filter															
11b		Overlap smoothing on a per macroblock basis based on OVERFLAGS															
	28:26	<p>PicType (Picture Type) This field specifies the coding type of the picture according to the Frame Coding Mode. When FCM = 00 01 (a Progressive or Interlaced Frame Picture): 000 = I001 = P010 = B011 = BI100 = Skipped Other encodings are reserved When FCM = 10 11 (a Field Picture) 000 = I/I001 = I/P010 = P/I011 = P/P100 = B/B101 = B/BI110 = BI/B111 = BI/BIA Although, for a field picture, it is set for a field-pair, but HW will only look at one field state only, and the other field state is don't care. This field is read and qualified with the SecondField flag internally. This field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For VC1 IT mode, driver needs to convert the interface to intel HW VLD Long Format interface.</p>															
	25:24	<p>FCM (Frame Coding Mode)</p>															

MFD_VC1_LONG_PIC_STATE

		<p>This is the same as the variable FCM defined in VC1. This field must be set to 0 for Simple and Main Profiles. This field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For VC1 IT mode, driver needs to convert the interface to intel HW VLD Long Format interface.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Disable</td> <td>Progressive Frame Picture</td> </tr> <tr> <td>01b</td> <td>Enable</td> <td>Interlaced Frame Picture</td> </tr> <tr> <td>10b</td> <td></td> <td>Field Picture with Top Field First</td> </tr> <tr> <td>11b</td> <td></td> <td>Field Picture with Bottom Field First</td> </tr> </tbody> </table>		Value	Name	Description	00b	Disable	Progressive Frame Picture	01b	Enable	Interlaced Frame Picture	10b		Field Picture with Top Field First	11b		Field Picture with Bottom Field First
Value	Name	Description																
00b	Disable	Progressive Frame Picture																
01b	Enable	Interlaced Frame Picture																
10b		Field Picture with Top Field First																
11b		Field Picture with Bottom Field First																
	23:21	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ													
Format:	MBZ																	
	20:16	<p>AltPQuant (Alternative Picture Quantization Value)</p> <p>This field is identical to the variable ALTPQUANT which is derived from VOPDQUANT configuration in the VC1 standard. This field must be set to 0 for Simple/Main I and BI pictures as VOPDQUANT is not present. This field is used in intel VC1 VLD Long Format mode only, not used in VC1 VLD and IT modes.</p>																
	15:13	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ													
Format:	MBZ																	
	12:8	<p>PQuant (Picture Quantization Value)</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>U5</td> </tr> </table> <p>This is the same as the calculated variable PQUANT in VC1 standard where $PQuant = PQINDEX$, except when $QUANTIZER = 0$ and $PQINDEX > 8$, it is given as $PQuant = (PQINDEX < 29) ? PQINDEX - 3 : PQINDEX * 2 - 31$. This field is used in all picture types (I, P, B and BI) and all operating modes (IT mode and intel and VLD modes).</p>		Format:	U5													
Format:	U5																	
	7:0	<p>BScaleFactor</p> <p>BScaleFactor This field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFRACTION in the VC1 standard, section 8.4.5.4. There are only 21 valid values corresponding to the 21 encodings of BFRACTION as shown in the table here. Other values are reserved. MSB of this field can be used to determine if BFRACTION is greater than or equal to 1/2, which is used to determine Motion Prediction Type for B pictures. Effectively, condition "$BFRACTION \geq 1/2$" is equivalent to condition "$BScaleFactor \geq 128$". This field is only valid for B pictures. This field is used only in intel VC1 VLD Long format mode, it is not used in VC1 VLD and IT modes.</p> <p>BFRACTION V L C BFRACTION B ScaleFactor 0001/21280011/3850102/31700111/4641003/41921011/ 5511102/510211100003/515311100014/520411100101/64311100115/621511101001/737111010 12/77411101103/711111101114/714811110005/718511110016/722211110101/83211110113/89 611111005/816011111017/8224</p>																
4	31:30	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ													
	Format:	MBZ																
29:28	<p>UnifiedMvMode (Unified Motion Vector Mode)</p> <p>This field is a combination of the variables MVMODE and MVMODE2 in the VC1 standard, for parsing Luma MVD from the bitstream. This field is used to signal 1MV vs 4MV allowed (Mixed</p>																	

MFD_VC1_LONG_PIC_STATE

Mode). This field is also used to signal Q-pel or Half-pel MVD read from the bitstream. The bicubic or bilinear Luma MC interpolation mode is duplicate information from Motion Vector Mode field, and is ignored here. This field is used in intel VC1 VLD Long Format mode only, it is not used in VC1 VLD and IT modes.

Value	Name	Description
00b		Mixed MV, Q-pel bicubic
01b		1-MV, Q-pel bicubic
10b		1-MV half-pel bicubic
11b		1-MV half-pel bilinear

27 FourMvSwitch (Four Motion Vector Switch)
 This field indicates if 4-MV is present for an interlaced frame P picture. It is identical to the variable 4MVS SWITCH (4 Motion Vector Switch) in VC1 standard. This field is used in intel VC1 VLD Long Format mode only, it is not used in VC1 VLD and IT modes.

Value	Name	Description
0h	Disable	only 1-MV
1h	Enable	1, 2, or 4 MVs

26 FastUVMCFlag (Fast UV Motion Compensation Flag)
 This field specifies whether the motion vectors for UV is rounded to half or full pel position. It is identical to the variable FASTUVMC in VC1 standard. This field is used in both VLD and IT modes. It is derived from $FASTUVMC = (bPicSpatialResid8 \gg 4) \& 1$ in both VLD and IT modes, and should have the same value as Motion Vector Model SBit.

Value	Name	Description
0h		no rounding
1h		quarter-pel offsets to half/full pel positions

25 RefFieldPicPolarity (Reference Field Picture Polarity)
 This field specifies the polarity of the one reference field picture used for a field P picture. It is derived from the variable REFFIELD defined in VC1 standard and is only valid when one field is referenced (NUMREF = 0) for a field P picture. When NUMREF = 0 and REFFIELD = 0, this field is the polarity of the reference I/P field that is temporally closest; When NUMREF = 0 and REFFIELD = 1, this field is the polarity of the reference I/P field that is the second most temporally closest. The distance is measured based on display order but ignoring the repeated field if present (due to RFF = 1). This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		Top (even) field
1h		Bottom (odd) field

24 NumRef (Number of References)
 This field indicates how many reference fields are referenced by the current (field) picture. It is identical to the variable NUMREF in the VC1 standard. This field is only valid for field P picture (FCM = 10 | 11). This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

MFD_VC1_LONG_PIC_STATE

	Value	Name	Description
	0h		One field referenced
	1h		Two fields referenced
23:20	BwdRefDist (Reference Distance) This field is valid only in B field pictures giving the value of BRFD. The field is ignored in P Picture. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.		
19:16	FwdRefDist (Reference Distance) Format: U4 This field is the number of frames between the current frame and its reference frame. It is derived from the syntax element REFDIST (P Reference Distance) in the VC1 standard. 0 means that the previous frame is the reference frame. It has the same value as of FRFD for both P and B field pictures. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.		
	Value	Name	
	[0, 15]		
15:12	Reserved Format: MBZ		
11:10	ExtendedDMVRange (Extended Differential Motion Vector Range Flag) This field specifies the differential motion vector range in interlaced pictures. It is equivalent to the variable DMVRANGE in the VC1 standard. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.		
	Value	Name	Description
	00b		No extended range
	01b		Extended horizontally
	10b		Extended vertically
	11b		Extended in both directions
9:8	ExtendedMVRRange (Extended Motion Vector Range Flag) This field specifies the motion vector range in quarter-pel or half-pel modes. It is equivalent to the variable MVRANGE in the VC1 standard. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes		
	Value	Name	Description
	00b		[-256, 255] x [-128, 127]
	01b		512, 511] x [-256, 255]
	10b		[-2048, 2047] x [-1024, 1023]
	11b		[-4096, 4095] x [-2048, 2047]
7:4	AltPQuantEdgeMask (Alternative Picture Quantization Edge Mask) This field is a bit mask for the four edges in clock-wise order, indicating whether AltPQuant is used for the edge macroblocks. It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard,		

MFD_VC1_LONG_PIC_STATE

		as shown in Error! Reference source not found..This field is valid only if AltPQuantConfig is 01. Bit 0: Left picture edge macroblocksBit 1: Top picture edge macroblocksBit 2: Right picture edge macroblocksBit 3: Bottom picture edge macroblocksThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.															
	3:2	<p>AltPQuantConfig (Alternative Picture Quantization Configuration)</p> <p>This field specifies the way AltPQuant is used in the picture. It determines how to compute the macroblock quantizer step size, MQANT. It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard, as shown in Error! Reference source not found..This field is unique to intel VC1 VLD Long format mode, and is not used in IT and modes.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td></td> <td>AltPQuant not used</td> </tr> <tr> <td>01b</td> <td></td> <td>AltPQuant is used and applied to edge macroblocks only</td> </tr> <tr> <td>10b</td> <td></td> <td>MQANT is encoded in macroblock layer</td> </tr> <tr> <td>11b</td> <td></td> <td>AltPQuant and PQuant are selected on macroblock basis</td> </tr> </tbody> </table>	Value	Name	Description	00b		AltPQuant not used	01b		AltPQuant is used and applied to edge macroblocks only	10b		MQANT is encoded in macroblock layer	11b		AltPQuant and PQuant are selected on macroblock basis
Value	Name	Description															
00b		AltPQuant not used															
01b		AltPQuant is used and applied to edge macroblocks only															
10b		MQANT is encoded in macroblock layer															
11b		AltPQuant and PQuant are selected on macroblock basis															
	1	<p>HalfQP</p> <p>This field is used for inverse quantization of AC coefficients. It is valid only when PQuant is used.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>															
	0	<p>PQuantUniform</p> <p>Indicating if uniform quantization applies to the picture. It is used for inverse quantization of the AC coefficients.QUANTIZER=001123PQUANTIZER --01--PQINDEX>=9<=8----</p> <p>PQuantUniform=010201ImplicitQuantizer = 0, andPQuantUniform = 0 is used to represent 2 cases : 1) QUANTIZER=01 andPQUANTIZER=0; and 2) QUANTIZER = 10b.ImplicitQuantizer = 0, and PQuantUniform = 1 is used to represent 2 cases : 1) QUANTIZER=01 and PQUANTIZER=1; and 2)QUANTIZER = 11bThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td></td> <td>Non-uniform</td> </tr> <tr> <td>1h</td> <td></td> <td>Uniform</td> </tr> </tbody> </table>	Value	Name	Description	0h		Non-uniform	1h		Uniform						
Value	Name	Description															
0h		Non-uniform															
1h		Uniform															
5	31	<p>BitplanePresentFlag (Bitplane Buffer Present Flag)</p> <p>This field indicates whether the bitplane buffer is present for the picture. If set, at least one of the fields listed in bits 22:16 is coded in non-raw mode, and Bitplane Buffer Base Address field in the VC1_BSD_BUF_BASE_STATE command points to the bitplane buffer. Otherwise, all the fields that are applicable for the current picture in bits 22:16 must be coded in raw mode.This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td></td> <td>bitplane buffer is not present</td> </tr> <tr> <td>1h</td> <td></td> <td>bitplane buffer is present</td> </tr> </tbody> </table>	Value	Name	Description	0h		bitplane buffer is not present	1h		bitplane buffer is present						
Value	Name	Description															
0h		bitplane buffer is not present															
1h		bitplane buffer is present															
	30	<p>ForwardMbRaw</p> <p>This field indicates whether the FORWARDMB field is coded in raw or non-raw mode.This field is only valid when PictureType is B.This field is unique to intel VC1 VLD Long format mode, and is</p>															

MFD_VC1_LONG_PIC_STATE

MFD_VC1_LONG_PIC_STATE		
	not used in IT and VC1 modes.	
	Value	Name
	0h	non-raw mode
	1h	raw mode
29	<p>MvTypeMbRaw</p> <p>This field indicates whether the MVTYPREMB field is coded in raw or non-raw mode. This field is only valid when PictureType is P. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>	
	Value	Description
	0h	Non-Raw Mode
	1h	Raw Mode
28	<p>SkipMbRaw</p> <p>This field indicates whether the SKIPMB field is coded in raw or non-raw mode. This field is only valid when PictureType is P or B. 0 = non-raw mode 1 = raw mode This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>	
	Value	Description
	0h	Disable Non-Raw Mode
	1h	Enable Raw Mode
27	<p>DirectMbRaw</p> <p>This field indicates whether the DIRECTMB field is coded in raw or non-raw mode. This field is only valid when PictureType is P or B. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>	
	Value	Description
	0h	Non-Raw Mode
	1h	Raw Mode
26	<p>OverflagsRaw</p> <p>This field indicates whether the OVERFLAGS field is coded in raw or non-raw mode. This field is only valid when PictureType is I or BI. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>	
	Value	Description
	0h	Non-Raw Mode
	1h	Raw Mode
25	<p>AcPredRaw</p> <p>This field indicates whether the ACPRED field is coded in raw or non-raw mode. This field is only valid when PictureType is I or BI. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>	
	Value	Description
	0h	Disable Non-Raw Mode
	1h	Enable Raw Mode

MFD_VC1_LONG_PIC_STATE

24	<p>FieldTxRaw</p> <p>This field indicates whether the FIELDTX field is coded in raw or non-raw mode. This field is only valid when PictureType is I or BI. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">Disable</td> <td style="text-align: center;">Non-Raw Mode</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">Enable</td> <td style="text-align: center;">Raw Mode</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Non-Raw Mode	1h	Enable	Raw Mode
Value	Name	Description								
0h	Disable	Non-Raw Mode								
1h	Enable	Raw Mode								
23	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
22:20	<p>MvTab (Motion Vector Table)</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">U3</td> </tr> </table> <p>This field specifies which motion vector table(s) is (are) used for motion vector (differential) decoding in a P or B picture. This field is the combination of the variables MVTAB and IMVTAB in the VC1 standard. Two bits are defined for progressive frame pictures; And two or three bits are defined for interlaced field/frame pictures depending on NUMREF and P or B picture types. This field is valid for P and B pictures. It is not valid for I pictures. For P or B progressive frame pictures 0 = Motion Vector Differential VLD Table 01 = Motion Vector Differential VLD Table 12 = Motion Vector Differential VLD Table 23 = Motion Vector Differential VLD Table 3. The other encodings are reserved. For P interlace field pictures with NUMREF = 0 or P/B interlace frame pictures 0 = 1-Reference Table 01 = 1-Reference Table 12 = 1-Reference Table 23 = 1-Reference Table 3. The other encodings are reserved. For P interlace field picture with NUMREF = 1 or B interlaced field pictures 0 = 2-Reference Table 01 = 2-Reference Table 12 = 2-Reference Table 23 = 2-Reference Table 34 = 2-Reference Table 45 = 2-Reference Table 56 = 2-Reference Table 67 = 2-Reference Table 7. The other encodings are reserved. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>	Format:	U3							
Format:	U3									
19:18	<p>FourMvBpTab (4-MV Block Pattern Table)</p> <p>This field specifies which table is used to decode the 4-MV block pattern (4MVBP) syntax element in 4-MV macroblocks. It is identical to the variables 4MVBPTAB in the VC1 standard, section 9.1.1.37. This field is valid only in interlace frame P, B pictures, or interlace field P, B pictures. It is not valid for I picture. For interlace field P and B pictures, it is only valid if UnifiedMvMode is equal to Mixed-MV Type. For interlace frame P picture, it is only valid if FourMvSwitch is 1. For interlace frame B picture, it is always valid. 0 = 4MVBP Table 01 = 4MVBP Table 12 = 4MVBP Table 23 = 4MVBP Table 3. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>									
17:16	<p>TwoMvBpTab (2MV Block Pattern Table)</p> <p>This field specifies which table is used to decode the 2MV block pattern (2MVBP) syntax element in 2MV field macroblocks. It is identical to the variables 2MVBPTAB in the VC1 standard, section 9.1.1.36. This field is valid only in interlace frame P/B pictures. It is not valid for I picture, nor for interlace field P or B pictures. 0 = 2MVBP Table 01 = 2MVBP Table 12 = 2MVBP Table 23 = 2MVBP Table 3. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>									
15:14	<p>Reserved</p>									

MFD_VC1_LONG_PIC_STATE

	Format:	MBZ									
13:12	TransType (Picture-level Transform Type) <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">U2</td> </tr> </table> <p>This field specifies the Transform Type at picture level. It is identical to the variable TTFRM in the VC1 standard, section 7.1.1.41. This field is only valid when TransTypeMbFlag is 1. Otherwise, it is reserved and MBZ. This field is set to 00 when VSTRANSFORM is 0 in the entry point layer. 00 = 8x8 Transform 01 = 8x4 Transform 10 = 4x8 Transform 11 = 4x4 Transform This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>		Format:	U2							
Format:	U2										
11	TransTypeMbFlag (Macroblock Transform Type Flag) <p>This field indicates whether Transform Type is fixed at picture level or variable at macroblock level. It is identical to the variable TTMBF in the VC1 standard, section 7.1.1.40. This field is set to 1 when VSTRANSFORM is 0 in the entry point layer. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #D9E1F2;"> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td></td> <td>variable transform type in macroblock layer</td> </tr> <tr> <td>1h</td> <td></td> <td>use picture level transform type TransType</td> </tr> </tbody> </table>		Value	Name	Description	0h		variable transform type in macroblock layer	1h		use picture level transform type TransType
Value	Name	Description									
0h		variable transform type in macroblock layer									
1h		use picture level transform type TransType									
10:8	MbModeTab (Macroblock Mode Table) <p>This field signals which code table is used to decode the macroblock mode syntax element (MBMODE) in the macroblock layer in a P or B picture. This field is identical to the variables MBMODETAB in the VC1 standard, section 9.1.1.33. This field is valid for interlace frame P, B picture and interlace field P, B picture. It is not valid for I picture, nor progressive frame P, B pictures. Two bits are defined for interlace frame P, B pictures; And three bits are defined for interlaced field P, B pictures. Two bits are defined for interlace frame P, B pictures. There are two set of code tables selected based on if UnifiedMvMode is equal to 4-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 30 Other encodings are invalid Three bits are defined for interlace field P, B pictures. There are two set of code tables selected based on if UnifiedMvMode is equal to Mixed-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 34 = Code Table 45 = Code Table 56 = Code Table 67 = Code Table 7 This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>										
7:6	TransAcY (Picture-level Transform Luma AC Coding Set Index, TRANSACTABLE2) BitFieldDesc										
5:4	TransAcUV (Picture-level Transform Chroma AC Coding Set Index, TRANSACTABLE) <p>This field, together with PQINDEX, specifies which intra AC coding set to be used for decoding the non-zero AC coefficients in a coded luma (Y) block. This field is the combination of the variables TRANSACFRM and TRANSACFRM2 in the VC1 standard. For I pictures, TransAcY is the same as TRANSACFRM2. For other pictures, it is the same as TRANSACFRM, and therefore must be programmed to be the same as TransAcUV. This field is valid for all picture types. 0 = Coding set index 01 = Coding set index 12 = Coding set index 23 is invalid This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>										
3	TransDcTab (Intra Transform DC Table) <p>This field specifies whether the low motion tables or the high motion tables are used to decode the Transform DC coefficients in intra-coded blocks. This field is identical to the variable</p>										

MFD_VC1_LONG_PIC_STATE											
		<p>TRANSDCTAB in the VC1 standard, section 8.1.1.2. This field is valid for all picture types. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td></td> <td>The high motion tables</td> </tr> <tr> <td style="text-align: center;">1h</td> <td></td> <td>The low motion tables</td> </tr> </tbody> </table>	Value	Name	Description	0h		The high motion tables	1h		The low motion tables
Value	Name	Description									
0h		The high motion tables									
1h		The low motion tables									
2:0		<p>CbpTab (Coded Block Pattern Table) This field specifies the table used to decode the CBPCY syntax element for each coded macroblock in P and B pictures. This field is combination of the variable CBPTAB for P and B frame pictures and the variable ICBPTAB in interlace field P, B pictures and interlace frame P, B pictures in the VC1 standard (Table 52 and Table 102). This field is reserved and MBZ for I or BI pictures as I only has a fixed table. 000 = Table 0 (Table 169 for P, B frames or Table 124 otherwise) 001 = Table 1 (Table 170 for P, B frames or Table 125 otherwise) 010 = Table 2 (Table 171 for P, B frames or Table 126 otherwise) 011 = Table 3 (Table 172 for P, B frames or Table 127 otherwise) 100 = Table 4 (Table 128 for interlace field/frame P, B pictures) 101 = Table 5 (Table 129 for interlace field/frame P, B pictures) 110 = Table 6 (Table 130 for interlace field/frame P, B pictures) 111 = Table 7 (Table 131 for interlace field/frame P, B pictures) This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.</p>									

MFD_VC1_SHORT_PIC_STATE

MFD_VC1_SHORT_PIC_STATE		
Source:		VideoCS
Length Bias:		2
DWord	Bit	Description
0	31:2	Command Type
	9	Default Value: 3h PARALLEL_VIDEO_PIPE
		Format: OpCode
	28:2	Pipeline
	7	Default Value: 2h MFD_VC1_SHORT_PIC_STATE
		Format: OpCode
	26:2	Media Command Opcode
	4	Default Value: 2h VC1_DEC
	Format: OpCode	
23:2		SubOpcode A
	1	Default Value: 1h
		Format: OpCode
20:1		SubOpcode B
	6	Default Value: 0h
		Format: OpCode
15:1		Reserved
	2	Format: MBZ
11:0		DWord Length
		Default Value: 0003h Excludes DWord (0,1)
		Format: =n
1	31:2	Reserved
	4	Format: MBZ
23:1		Picture Height
	6	Format: U8-1
<p>This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead). This field is used in VLD and IT modes. Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boundary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.</p>		

MFD_VC1_SHORT_PIC_STATE				
	15:8	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	7:0	Picture Width <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8-1</td> </tr> </table> <p>This field indicates the width of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureWidthInMBs equals 120 (1920 divided by 16). This field is used in VLD and IT modes.</p>	Format:	U8-1
Format:	U8-1			
2	31:2 4 Bitplane Buffer Pitch Minus 1 <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8-1</td> </tr> </table> <p>Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format, it is written by an application and later read by the HW. In VC1 Long Format, it is written by an application, and later read by the HW. But in VC1 Short Format, it is written and read by H/W only. This field is specified for better performance. The pitch must be equal to PictureWidthInMBs/2. For VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2. VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row. This field is not used in IT mode, used in VLD mode only. For VC1 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.</p>	Format:	U8-1	
Format:	U8-1			
	23	Interpolation Rounder Control Used only in MC operation. This field specifies the rounding control value used in interpolation operation of motion prediction process. Note: This bit field is taken from bRcontrol in PictureParameters data structure. This field is used in VLD and IT modes.		
	22:2 0	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
	19:1 6	Motion Vector Mode This field indicates one of the following motion compensation interpolation modes for P and B pictures. The MC interpolation modes apply to prediction values of luminance blocks and are always in quarter-sample. For chrominance blocks, it always performs bilinear interpolation with either half-pel or quarter-pel precision. 0XX0 = Chroma Quarter -pel + Luma bicubic. (can only be 1MV) 0XX1 = Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV) 1XX0 = Chroma Quarter -pel + Luma bilinear. (can only be 1MV) 1XX1 = Chroma Half-pel + Luma bilinear. Note: Bits 19:16 are taken from bMVprecisionAndChromaRelation in PictureParameters data structure. Bit 19 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MC. Bit 16 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion. This field is used in both VLD and IT modes. Before the polarity of Chroma Half-pel or Q-pel is reversed from Spec, now I have fixed it to match with VC1 Spec. ???		
	15	DmvSurfaceValid		

MFD_VC1_SHORT_PIC_STATE

		Indicated when the DMV read surface is valid. This surface stored the direct motion vectors. This field is set for B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture). This field is not used in IT mode, used in VLD mode only.
14:1	Reserved	
2	Format:	MBZ
11	VC1 Profile	specifies the bitstream profile. Note: This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not. This field is used in both VLD and IT modes.
	Value	Name
		Description
0h	[Default]	current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)
1h		current picture is in Advanced Profile
10:6	Reserved	
	Format:	MBZ
5	Backward Prediction Present Flag	Note : a B picture that only uses forward prediction may have this flag set to 1 as well. Driver may still need to provide a valid reference picture index. This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicBackwardPrediction in VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in VC1 VLD and IT mode.
4	Intra Picture Flag	This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicIntra in VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in VC1 VLD and IT mode.
	Value	Name
		Description
0h		entire picture can have a mixture of intra and inter MB type or just inter MB type.
1h		entire picture is coded in intra MB type
3	SecondField	This flag is set for the second field in field pictures. This field is used in both VLD and IT modes.
2	Reserved	
	Format:	MBZ
1:0	Picture Structure	This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicStructure in VC1 spec. The Picture Structure and Progressive Pic Type are used to derive the picture structure as specified in FCM, in VC1 VLD and IT mode.

MFD_VC1_SHORT_PIC_STATE																	
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>01b</td> <td></td> <td>top field (bit 0)</td> </tr> <tr> <td>10b</td> <td></td> <td>bottom field (bit 1)</td> </tr> <tr> <td>11b</td> <td></td> <td>frame (both fields are present)</td> </tr> <tr> <td>00b</td> <td></td> <td>illegal</td> </tr> </tbody> </table>	Value	Name	Description	01b		top field (bit 0)	10b		bottom field (bit 1)	11b		frame (both fields are present)	00b		illegal
Value	Name	Description															
01b		top field (bit 0)															
10b		bottom field (bit 1)															
11b		frame (both fields are present)															
00b		illegal															
3	31	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
	30	<p>Overlap Smoothing Enable Flag</p> <p>This field is the decoded syntax element OVERLAP in bitstream. Indicates if Overlap smoothing is ON at the picture level. This field is used in both VLD and IT modes.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>to disable overlap smoothing filter</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>to enable overlap smoothing filter</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	to disable overlap smoothing filter	1h	Enable	to enable overlap smoothing filter						
	Value	Name	Description														
	0h	Disable	to disable overlap smoothing filter														
1h	Enable	to enable overlap smoothing filter															
29	<p>Range Reduction Scale</p> <table border="1"> <tr> <td>Access:</td> <td>None</td> </tr> </table> <p>This field specifies whether the reference picture pixel values should be scaled up or scaled down on-the-fly, if RangeReduction is Enabled. NOTE: This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture only. Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable [Default]</td> <td>Scale down reference picture by factor of 2</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Scale up reference picture by factor of 2</td> </tr> </tbody> </table>	Access:	None	Value	Name	Description	0h	Disable [Default]	Scale down reference picture by factor of 2	1h	Enable	Scale up reference picture by factor of 2					
Access:	None																
Value	Name	Description															
0h	Disable [Default]	Scale down reference picture by factor of 2															
1h	Enable	Scale up reference picture by factor of 2															
28	<p>Range Reduction Enable</p> <p>This field specifies whether on-the-fly pixel value range reduction should be performed for the preceding (or forward) reference picture. Along with RangeReductionScale to specify whether scale up or down should be performed. It is not the same value as RANGEREDFRM Syntax Element (PictureParameters bPicDeblocked bit 5) in the Picture Header. This field is for Main Profile only. Simple Profile is always disable, and not applicable to Advanced Profile. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this field represents the state of the forward/preceding reference picture only. Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.</p>																

MFD_VC1_SHORT_PIC_STATE			
	Value	Name	Description
	0h	Disable [Default]	Range reduction is not performed
	1h	Enable	Range reduction is performed
27:2	Reserved		
4	Format:		MBZ
23:2	Progressive Pic Type		
2	This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicExtrapolation in VC1 spec. The Picture Structure and Progressive Pic Type are used to derive the picture structure as specified in FCM, in VC1 VLD and IT mode.		
	Value	Name	Description
	0		progressive only picture
	1		progressive only picture
	2		interlace picture (frame-interlace or field-interlace)
	3		illegal
21	Reserved		
	Format:		MBZ
20:1	P-Pic Ref Distance		
6	Access:		None
	This element defines the number of frames between the current frame and the reference frame. It is the same as the REFDIST SE in VC1 interlaced field picture header. It is present if the entry-level flag REFDIST_FLAG == 1, and if the picture type is not one of the following types: B/B, B/BI, BI/B, BI/BI. If the entry level flag REFDIST_FLAG == 0, REFDIST shall be set to the default value of 0. This field is used in VC1 VLD mode only, not used in IT and intel VC1 VLD Long Format modes.		
	Value	Name	
	0-16		unsigned integer
	0h		[Default]
15:1	QUANTIZER		
4	Value	Name	Description
	00b		implicit quantizer at frame level
	01b		explicit quantizer at frame level, and use PQUANTIZER SE to specify uniform or non-uniform
	10b		explicit quantizer, and non-uniform quantizer for all frames
	11b		explicit quantizer, and uniform quantizer for all frames
13	MULTIRES Present Flag (for Simple/Main Profile only)		
	Value	Name	Description
	0h		RESPIC Parameter is present in the picture header
	1h		RESPIC Parameter is present in the picture header

MFD_VC1_SHORT_PIC_STATE			
12	SYNCMARKER Present Flag (for Simple/Main Profile only)		
	Value	Name	
	Description		
	0		Bitstream for Simple and Main Profile has no sync marker
	1		Bitstream for Simple and Main Profile may have sync marker(s)
	11	RANGERED Present Flag (for Simple/Main Profile only)	
		It is needed for Picture Header Parsing. Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.	
		Value	Name
		Description	
	0		Range Reduction Parameter (RANGEREDFRM) is not present in the picture header
	1		Range Reduction Parameter (RANGEREDFRM) is present in the picture header.
	10:8	MAXBFRAMES	
Number of consecutive B Frames.			
7	PANSCAN Present Flag		
	Value	Name	
	Description		
	0		Pan Scan Parameters are not present in the picture header
1		Pan Scan Parameters are present in the picture header	
6	REFDIST_FLAG		
For header parsing REFDIST. This is used in VC1 VLD mode only, not used in IT and intel VC1 VLD modes.			
5	Reserved		
4	FastUVMCFlag (Fast UV Motion Compensation Flag)		
	This field specifies whether the motion vectors for UV is rounded to half or full pel position. It is identical to the variable FASTUVMC in VC1 standard. This field is used in both VLD and IT modes. It is derived from $FASTUVMC = (bPicSpatialResid8 \gg 4) \& 1$ in both VLD and IT modes, and should have the same value as Motion Vector ModeLSBit.		
	Value	Name	
	Description		
	0h		no rounding
1h		quarter-pel offsets to half/full pel positions	
3	EXTENDED_MV Present Flag		
	BitFieldDesc		
	Value	Name	
	Description		
0h		Extended_MV is not present in the picture header	
1h		Extended_MV is present in the picture header	
2:1	DQUANT		
	Access:	None	
	Format:	U2	
	Use for Picture Header Parsing of VOPDUANT elements		
	Value	Name	
Description			

MFD_VC1_SHORT_PIC_STATE

		0h	[Default]	
		00b		no VOPDQUANT elements; Quantizer cannot vary in frame, same quantization step size PQUANT is used for all MBs in the frame
		01b		refer to VC1 Spec. for all the MB position dependent quantizer selection
		10b		The macroblocks located on the picture edge boundary shall be quantized with ALTPQUANT while the rest of the macroblocks shall be quantized with PQUANT.
		11b	Reserved	
	0	VSTRANSFORM flag		
		Value	Name	Description
		0h	Disable	variable-sized transform coding is not enabled
		1h	Enable	variable-sized transform coding is enabled
4	31:2	Reserved		
	9	Format:		MBZ
	28:2	BFraction Enumeration		
	4	<p>This field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFACTION in the VC1 standard, section 8.4.5.4. There are only 21 valid values corresponding to the 21 encodings of BFACTION as shown in the table here. Other values are reserved. The VLD decoded value of BFACTION (from the picture header) is mapped into an enum value from 0 to 20. (??? MSB of this field can be used to determine if BFACTION is greater than or equal to 1/2, which is used to determine Motion Prediction Type for B pictures. Effectively, condition "BFACTION >= 1/2" is equivalent to condition "ScaleFactor >= 128". ??? How can the enum replicate this feature ???) This field is only valid for B pictures. This field is used only in VC1 VLD mode, it is not used in Intel VC1 VLD Long Format mode and IT mode. BFACTION_VLC_BFACTION_Enum 0001/200011/310102/320111/431003/441011/551102/5611100003/5711100014/5811100101/6911100115/61011101001/71111101012/71211101103/71311101114/71411110005/71511110016/71611110101/81711110113/81811111005/81911111017/82011111111 BIPic Indicator 31 (optional)</p>		
	23:9	Reserved		
		Format:		MBZ
	8	4MV Allowed Flag		
	7	POSTPROC Flag		
	6	PULLDOWN		
	5	INTERLACE		
	4	TFCNTRFLAG		
	3	FINTERFLAG		
	2	REFPIC Flag		
		For a BI picture, REFPIC flag must set to 0. For I and P picture, REFPIC flag must set to 0. For a B picture, REFPIC flag must set to 0, except for a B-field in interlaced field mode which can be 0		

MFD_VC1_SHORT_PIC_STATE

or 1 (e.g. the top B field can be used as a reference for decoding its corresponding bottom B-field in a field pair). In VLD mode, this flag cannot be used as an optimization signaling for an I or P picture that is not used as a reference picture. This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicDeblockConfined[bit2] in VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in VC1 VLD and IT mode.

Value	Name	Description
0h		the current picture after decoded, will never used as a reference picture
1h		the current picture after decoded, will be used as a reference picture later

1 **PSF**

0 **EXTENDED_DMV Present Flag**

Value	Name	Description
0h	[Default]	Extended_DMV is not present in the picture header
1h		Extended_DMV is present in the picture header

MFD_VP8_BSD_OBJECT

MFD_VP8_BSD_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>The MFD_VP8_BSD_OBJECT command is the only primitive command for the VP8 Decoding Pipeline. The Partitions of the bitstream is loaded as indirect data object. Before issuing a MFD_VP8_BSD_OBJECT command, all VP8 frame level states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_VP8_BSD_OBJECT command. Context switch interrupt is not supported by this command.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFD_VP8_BSD_OBJECT
		Format:	OpCode
	26:24	Media Command OpCode	
		Default Value:	4h VP8_DEC
		Format:	OpCode
	23:21	subOpcodeA	
Default Value:		1h	
Format:		OpCode	
20:16	subOpcodeB		
	Default Value:	8h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	14h Excludes DWord (0,1)	
	Format:	=n	
1	31:21	Reserved	
		Format:	MBZ
	20:16	Partition0 CPBAC Entropy Count	Pass the Partition0 CPBAC State to HW. Max value is 24.
		Partition0 CPBAC Entropy Range	Pass the Partition0 CPBAC State to HW.
	7:6	Reserved	

MFD_VP8_BSD_OBJECT			
		Format: MBZ	
5:4		Coded Num of Coeff Token Partitions Num of Partitions = $2^{\text{CodedNumCoeffTokenPartitions}}$. 0 = 1 Partition only 1 = 2 Partitions 2 = 4 Partitions 3 = 8 Partitions are present in the bitstream.	
3		Reserved Format: MBZ	
2:0		Partition0 FirstMBBitOffset from Frame Header Allow HW to jump to the location in the bitstream where per MB information starts in the Partition0.	
2	31:24	Partition0 CPBAC Entropy Value Pass the Partition0 CPBAC State to HW.	
	23:0	Reserved Format: MBZ	
3	31:24	Reserved Format: MBZ	
	23:0	Indirect Partition0 Data Length This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. <table border="1" style="width: 100%; margin-top: 10px;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>This needs to be set to the (actual Partition 0 length + 1) in bytes</td> </tr> </table>	Programming Notes
Programming Notes			
This needs to be set to the (actual Partition 0 length + 1) in bytes			
4	31:0	Indirect Partition0 Data Start Offset This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.	
5	31:24	Reserved Format: MBZ	
	23:0	Indirect Partition1 Data Length This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. <table border="1" style="width: 100%; margin-top: 10px;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>This needs to be set to the (actual Partition 1 length + 1) in bytes</td> </tr> </table>	Programming Notes
Programming Notes			
This needs to be set to the (actual Partition 1 length + 1) in bytes			
6	31:0	Indirect Partition1 Data Start Offset This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores	

MFD_VP8_BSD_OBJECT				
		this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.		
7	31:24	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	23:0	Indirect Partition2 Data Length This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>This needs to be set to the (actual Partition 2 length + 1) in bytes</td> </tr> </table>	Programming Notes	This needs to be set to the (actual Partition 2 length + 1) in bytes
Programming Notes				
This needs to be set to the (actual Partition 2 length + 1) in bytes				
31:0	Indirect Partition2 Data Start Offset This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.			
9	31:24	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	23:0	Indirect Partition3 Data Length This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td>This needs to be set to the (actual Partition 3 length + 1) in bytes</td> </tr> </table>	Programming Notes	This needs to be set to the (actual Partition 3 length + 1) in bytes
Programming Notes				
This needs to be set to the (actual Partition 3 length + 1) in bytes				
31:0	Indirect Partition3 Data Start Offset This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.			
11	31:24	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
	23:0	Indirect Partition4 Data Length This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table>	Programming Notes	
Programming Notes				

MFD_VP8_BSD_OBJECT				
		This needs to be set to the (actual Partition 4 length + 1) in bytes		
12	31:0	<p>Indirect Partition4 Data Start Offset</p> <p>This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.</p>		
13	31:24	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
23:0	<p>Indirect Partition5 Data Length</p> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table> <p>This needs to be set to the (actual Partition 5 length + 1) in bytes</p>	Programming Notes		
Programming Notes				
14	31:0	<p>Indirect Partition5 Data Start Offset</p> <p>This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.</p>		
15	31:24	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
23:0	<p>Indirect Partition6 Data Length</p> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table> <p>This needs to be set to the (actual Partition 6 length + 1) in bytes</p>	Programming Notes		
Programming Notes				
16	31:0	<p>Indirect Partition6 Data Start Offset</p> <p>This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.</p>		
17	31:24	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
23:0	<p>Indirect Partition7 Data Length</p> <p>This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The</p>			

MFD_VP8_BSD_OBJECT											
		<p>Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: center; background-color: #e6f2ff;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">This needs to be set to the (actual Partition 7 length + 1) in bytes</td> </tr> </tbody> </table>	Programming Notes		This needs to be set to the (actual Partition 7 length + 1) in bytes						
Programming Notes											
This needs to be set to the (actual Partition 7 length + 1) in bytes											
18	31:0	<p>Indirect Partition7 Data Start Offset This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.</p>									
19	31:24	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
23:0	<p>Indirect Partition8 Data Length This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: center; background-color: #e6f2ff;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">This needs to be set to the (actual Partition 8 length + 1) in bytes</td> </tr> </tbody> </table>	Programming Notes		This needs to be set to the (actual Partition 8 length + 1) in bytes							
Programming Notes											
This needs to be set to the (actual Partition 8 length + 1) in bytes											
20	31:0	<p>Indirect Partition8 Data Start Offset This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.</p>									
21	31	<p>Concealment Method This field specifies the method used for concealment when error is detected.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Intra 16x16 Prediction</td> <td>A copy from the current picture is performed using Intra 16x16 Prediction method.</td> </tr> <tr> <td>1</td> <td>Inter P Copy</td> <td>A copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field.</td> </tr> </tbody> </table>	Value	Name	Description	0	Intra 16x16 Prediction	A copy from the current picture is performed using Intra 16x16 Prediction method.	1	Inter P Copy	A copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field.
	Value	Name	Description								
	0	Intra 16x16 Prediction	A copy from the current picture is performed using Intra 16x16 Prediction method.								
	1	Inter P Copy	A copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field.								
	30:18	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ										
17:16	<p>Conceal_Pic_Id (Concealment Picture ID)</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>[Concealment Method] == 1</td> </tr> </table> <p>This field identifies the picture in the reference list to be used for concealment. This field is only valid if Concealment Method is Inter P Copy. 00 - Last Decoded Picture 01 - Golden Reference Picture 02 - Alternate Reference Picture 03 - User provided Reference Picture</p>	Exists If:	[Concealment Method] == 1								
Exists If:	[Concealment Method] == 1										
15	<p>Reserved</p>										

MFD_VP8_BSD_OBJECT

	Format:	MBZ
14	BSDPrematureComplete Error Handling It occurs in situation where the decode is completed but there are still data in the bitstream.	
	Value	Name
	0	Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling
	1	Set the interrupt to the driver (provide MMIO registers for MB address R/W)
13	Reserved	
	Format:	MBZ
12	MPR Error (MV out of range) Handling	
	Value	Name
	0	Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling
	1	Set the interrupt to the driver (provide MMIO registers for MB address R/W)
11	Reserved	
	Format:	MBZ
10	Entropy Error Handling	
	Value	Name
	0	Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling
	1	Set the interrupt to the driver (provide MMIO registers for MB address R/W)
9	Reserved	
	Format:	MBZ
8	MB Header Error Handling	
	Value	Name
	0	Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling
	1	Set the interrupt to the driver (provide MMIO registers for MB address R/W)
7:0	Reserved	
	Format:	MBZ

MFX_AVC_DIRECTMODE_STATE

MFX_AVC_DIRECTMODE_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This is a picture level command and is issued once per picture. All DMV buffers are treated as standard media surfaces, in which the lower 6 bits are used for conveying surface states. Current Pic POC number is assumed to be available in POCList[32 and 33] of the MFX_AVC_DIRECTMODE_STATE Command. This command is only valid in the AVC decoding in VLD and IT modes, and AVC encoder mode. The same command supports both Long and Short AVC Interface. The DMV buffers are not required to be programmed for encoder mode.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_SINGLE_DW
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	1h AVC_COMMON
		Format:	OpCode
	23:21	SubOpcodeA	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpcodeB		
	Default Value:	2h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0045h Excludes DWord (0,1)	
	Format:	=n	
1..32	1023:0	Direct MV Buffer for Reference Frame 0 to 15 - Base Address	
		Format:	SplitBaseAddress64ByteAligned[16]
<p>This field is for the Pre-Deblocking Destination Address, and provides the base address of the DMV buffer for reference frames 2 to 31. They are needed if the current B-Picture has MBs coded in direct mode. This is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned. There are a total of 32 possible Direct MV Read Buffers (not including the current write buffer of the current picture) to read in the corresponding DMV. Each read buffer size is 557,056 bytes for 1 frame</p>			

MFX_AVC_DIRECTMODE_STATE						
		<p>(the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution). The adjacent DMV buffers are paired ([2 and 3], [4 and 5], [N and N+1], ..[30 and 31]).</p> <p>This field is changed to one per frame: both top and bottom field share the same Direct MV Buffer Base Address.</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="2">Programming Notes</td> </tr> <tr> <td colspan="2">This field is ignored if PreDeblockOutEnable is set to 0 (disable).</td> </tr> </table>	Programming Notes		This field is ignored if PreDeblockOutEnable is set to 0 (disable).	
Programming Notes						
This field is ignored if PreDeblockOutEnable is set to 0 (disable).						
33	31:0	<p>Direct MV Buffer for Reference Frame 0 to 15 - Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes		
Format:	MemoryAddressAttributes					
34..35	63:0	<p>Direct MV Buffer for Write - Base Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>SplitBaseAddress64ByteAligned</td> </tr> </table> <p>This field provides the base address of the DMV write-only buffer for the current decoding frame/field. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned, i.e. the same as the above DMV read/write buffers. These 2 buffers can only be addressed by $[img_dec_fs_idc[4:0] \ll 1 + img_structure[1]]$ for the current picture being decoded.</p> <p>Each write buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution).</p> <p>DMV write buffer 32 is valid only if the current picture is a progressive frame, MbAff frame, or a top field. DMV write buffer 33 is valid only if the current picture is a bottom field.</p> <p>GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example, GraphicsAddress[47:12] is a 4KB page address.</p>	Format:	SplitBaseAddress64ByteAligned		
Format:	SplitBaseAddress64ByteAligned					
36	31:0	<p>Direct MV Buffer for Write - Attributes</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes		
Format:	MemoryAddressAttributes					
37..70	1087:0	<p>POCList[34][31:0]</p> <p>Each POC value is a signed 32-bit number. One-to-one correspondence with the 34 Direct MV Buffer Address for Reference and Current Frames/Fields. There are 34 POC entries in the list. For reference picture, only the lower 32 POC [0-31] entries can be used, and POCList[] is indexed by the frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottom Field). For current picture, all 34 POC entries [0-33] can be addressed by POCList[$img_dec_fs_idc[4:0] \ll 1 + img_structure[1]$]. For frame-only mode, every other entry is skipped. For MBAFF and field-only picture, each entry is a field POC, and every two entries are paired.</p>				

MFX_AVC_IMG_STATE

MFX_AVC_IMG_STATE				
Source:	VideoCS			
Length Bias:	2			
This must be the very first command to issue after the surface state, the pipe select and base address setting commands. This command supports both Long and Short VLD and IT AVC Decoding Interface.				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	28:27	Pipeline	
			Default Value:	2h MFX_AVC_IMG_STATE
			Format:	OpCode
	26:24	26:24	Media Command Opcode	
			Default Value:	1h AVC_COMMON
			Format:	OpCode
	23:21	23:21	SubOpcode A	
Default Value:			0h	
Format:			OpCode	
20:16	20:16	SubOpcode B		
		Default Value:	0h	
		Format:	OpCode	
15:12	15:12	Reserved		
		Format:	MBZ	
11:0	11:0	DWord Length		
		Default Value:	0Ch Excludes DWord (0,1)	
		Format:	=n	
1	31:16	Reserved		
		Format:	MBZ	
15:0	15:0	Frame Size		
		Format:	U16-1	
<p>The value for FrameSizeInMBs must match the product of FrameWidthInMBs and FrameHeightInMBs, Max.</p> <p>Screen resolution is therefore limited to 256 x 256 in MB unit. It is enough to cover all the Profile-Level specified in the current HD-DVD specification. E.g.. for 1920x1080, FrameSizeInMBs[15:0] = 8160 (1920/16 * 1088/16; rounded up 1080.) This parameter is specified for Intel interface only.</p>				

MFX_AVC_IMG_STATE										
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,16383]</td> <td></td> <td>representing Number of MBs [1,16384]</td> </tr> </tbody> </table>	Value	Name	Description	[0,16383]		representing Number of MBs [1,16384]		
Value	Name	Description								
[0,16383]		representing Number of MBs [1,16384]								
2	31:24	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
	23:16	Frame Height <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>U8-1</td> </tr> </table> <p>It is set to the value of (FrameHeightInMBsMinus1+ 1). Since the max value for FrameHeightInMBs is 255, the max allowed value for FrameHeightInMBsMinus1 is only 254. The min value for FrameHeightInMBs is 1. Although the max. value that can be specified for FrameHeightInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameHeightInMBs must not exceed the max value of FrameSizeInMBs[14:0]. e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead). It is derived from $\text{FrameHeightInMbs} = (2 - \text{frame_mbs_only_flag}) * \text{PicHeightInMapUnits}$ and $\text{PicHeightInMbs} = \text{FrameHeightInMbs} / (1 + \text{field_pic_flag})$ internally done. For MBAFF, PicHeightInMapUnits is in MB pair unit, so the bitstream sends only half frame height.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,255]</td> <td></td> <td>representing height [1,256]</td> </tr> </tbody> </table>	Format:	U8-1	Value	Name	Description	[0,255]		representing height [1,256]
	Format:	U8-1								
	Value	Name	Description							
	[0,255]		representing height [1,256]							
15:8	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
7:0	Frame Width <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>U8-1</td> </tr> </table> <p>It is set to the value of (FrameWidthInMBsMinus1+ 1). Since the max value for FrameWidthInMBs is 255, the max allowed value for FrameWidthInMBsMinus1 is only 254. The min value for FrameWidthInMBs is 1. Although the max. value that can be specified for FrameWidthInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameWidthInMBs must not exceed the max value of FrameSizeInMBs[14:0]. e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead). It is derived from $\text{FrameWidthInMbs} = (2 - \text{frame_mbs_only_flag}) * \text{PicWidthInMapUnits}$ and $\text{PicWidthInMbs} = \text{FrameWidthInMbs} / (1 + \text{field_pic_flag})$ internally done. For MBAFF, PicWidthInMapUnits is in MB pair unit, so the bitstream sends only half frame width.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0,255]</td> <td></td> <td>representing width [1,256]</td> </tr> </tbody> </table>	Format:	U8-1	Value	Name	Description	[0,255]		representing width [1,256]	
Format:	U8-1									
Value	Name	Description								
[0,255]		representing width [1,256]								
3	31:29	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
28:24	Second Chroma QP Offset Signed integer value. It should be in the range of -12 to +12 (according to AVC spec). It specifies the offset for determining QP Cr from QP Y. It is set to the upper 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS. Chroma_qp_offset [4:0] - chroma_qp_offset_bits (from the current active									

MFX_AVC_IMG_STATE											
		PPS)Chroma_qp_offset [9:5] - second_chroma_qp_offset_bits									
23:21	Reserved	Format: MBZ									
20:16	First Chroma QP Offset	Signed integer value. It should be in the range of -12 to +12 (according to AVC spec).It specifies the offset for determining QP Cb from QP Y. It is set to the lower 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS.Chroma_qp_offset [4:0] - chroma_qp_offset_bits (from the current active PPS)Chroma_qp_offset [9:5] - second_chroma_qp_offset_bits									
15:14	Reserved	Format: MBZ									
13	RhoDomain Rate Control Enable	Format: Enable									
		This field indicates if RhoDomain related parameters are present in the MFX_AVC_IMAGE_STATE. (AverageMacroblockQP). It enables the Rho Domain statistics collection.									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>RhoDomain rate control parameters are not present in MFX_AVC_IMAGE_STATE</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>RhoDomain rate control parameters are present in MFX_AVC_IMAGE_STATE.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	RhoDomain rate control parameters are not present in MFX_AVC_IMAGE_STATE	1	Enable	RhoDomain rate control parameters are present in MFX_AVC_IMAGE_STATE.
Value	Name	Description									
0	Disable	RhoDomain rate control parameters are not present in MFX_AVC_IMAGE_STATE									
1	Enable	RhoDomain rate control parameters are present in MFX_AVC_IMAGE_STATE.									
		Programming Notes									
		This field must set to '0' for B pictures.									
12	Weighted_Pred_Flag	Format: Enable									
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable [Default]</td> <td>specifies that weighted prediction is not used for P and SP slices</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>specifies that weighted prediction is used for P and SP slices</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable [Default]	specifies that weighted prediction is not used for P and SP slices	1	Enable	specifies that weighted prediction is used for P and SP slices
Value	Name	Description									
0	Disable [Default]	specifies that weighted prediction is not used for P and SP slices									
1	Enable	specifies that weighted prediction is used for P and SP slices									
		Programming Notes									
		This field must set to '0' for B and I pictures.									
11:10	Weighted_BiPred_Idx	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>DEFAULT [Default]</td> <td>Specifies that the default weighted prediction is used for B slices</td> </tr> <tr> <td>1</td> <td>EXPLICIT</td> <td>Specifies that explicit weighted prediction is used for B</td> </tr> </tbody> </table>	Value	Name	Description	0	DEFAULT [Default]	Specifies that the default weighted prediction is used for B slices	1	EXPLICIT	Specifies that explicit weighted prediction is used for B
Value	Name	Description									
0	DEFAULT [Default]	Specifies that the default weighted prediction is used for B slices									
1	EXPLICIT	Specifies that explicit weighted prediction is used for B									

MFX_AVC_IMG_STATE													
		<table border="1"> <tr> <td></td> <td></td> <td>slices</td> </tr> <tr> <td>2</td> <td>IMPLICIT</td> <td>Specifies that implicit weighted prediction is used for B slices.</td> </tr> <tr> <td>3</td> <td>Reserved</td> <td>Illegal value</td> </tr> </table>			slices	2	IMPLICIT	Specifies that implicit weighted prediction is used for B slices.	3	Reserved	Illegal value		
				slices									
		2	IMPLICIT	Specifies that implicit weighted prediction is used for B slices.									
		3	Reserved	Illegal value									
	Programming Notes												
	This field must set to 0 for P and I pictures.												
	9:8	ImgStruct - Image Structure, img_structure[1:0] The current encoding picture structure can only takes on 3 possible values <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Frame Picture</td> </tr> <tr> <td>01b</td> <td>Top Field Picture</td> </tr> <tr> <td>11b</td> <td>Bottom Field Picture</td> </tr> <tr> <td>10b</td> <td>Invalid, not allowed.</td> </tr> </tbody> </table>		Value	Name	00b	Frame Picture	01b	Top Field Picture	11b	Bottom Field Picture	10b	Invalid, not allowed.
	Value	Name											
	00b	Frame Picture											
	01b	Top Field Picture											
11b	Bottom Field Picture												
10b	Invalid, not allowed.												
Programming Notes													
img_structure[0] can be used as a flag to distinguish between frame and field structure. It must be consistent with the field_pic_flag setting in the Slice Header. This parameter is specified for Intel interface only, as a separate state (instead the img_structure[1] is embedded inside the picture definition).													
7:0	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ									
Format:	MBZ												
4	31:16	MinFrameWSize <table border="1"> <tr> <td>Default Value:</td> <td>0h</td> </tr> <tr> <td>Format:</td> <td>U16</td> </tr> </table> <p>Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax (DWORD 10 bits 29:16). This field is reserved in Decode mode.</p> <p>The programmable range $02^{18}-1$ When MinFrameWSizeUnits is 00. Programmable range is $02^{20}-1$ when MinFrameWSizeUnits is 01. Programmable range is $02^{26}-1$ when MinFrameWSizeUnits is 10. Programmable range is $02^{32}-1$ when MinFrameWSizeUnits is 11.</p>	Default Value:	0h	Format:	U16							
		Default Value:	0h										
	Format:	U16											
15	MbStatEnabled <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> Enable reading in MB status buffer (a.k.a. encoding stream-out buffer) Note: For multi-	Format:	Enable										
Format:	Enable												

MFX_AVC_IMG_STATE

		<p>pass encoder, all passes except the first one need to set this value to 1. By setting the first pass to 0, it does save some memory bandwidth.</p> <p>In VDenc mode this must be set to zero as no MB level rate control is used.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>Disable Reading of Macroblock Status Buffer</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>Enable Reading of Macroblock Status Buffer</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Disable Reading of Macroblock Status Buffer	1	Enable	Enable Reading of Macroblock Status Buffer						
Value	Name	Description															
0	Disable	Disable Reading of Macroblock Status Buffer															
1	Enable	Enable Reading of Macroblock Status Buffer															
14	LoadSlicePointerFlag	<p>Format: Enable</p> <p>LoadBitStreamPointerPerSlice (Encoder-only)To support multiple slice picture and additional header/data insertion before and after an encoded slice.When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream.When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>Load BitStream Pointer only once for the first slice of a frame</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Load BitStream Pointer only once for the first slice of a frame	1	Enable	Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field						
Value	Name	Description															
0	Disable	Load BitStream Pointer only once for the first slice of a frame															
1	Enable	Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field															
13	Reserved																
12	MvUnpackedFlag	<p>MVUnPackedEnable (Encoder Only)This field is reserved in Decode mode.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>PACKED</td> <td>use packed MV format</td> </tr> <tr> <td>1</td> <td>UNPACKED</td> <td>use unpacked 8MV/32MV format only</td> </tr> </tbody> </table>	Value	Name	Description	0	PACKED	use packed MV format	1	UNPACKED	use unpacked 8MV/32MV format only						
Value	Name	Description															
0	PACKED	use packed MV format															
1	UNPACKED	use unpacked 8MV/32MV format only															
11:10	ChromaFormatIdc	<p>Chroma Format IDC, ChromaFormatIdc[1:0]It specifies the sampling of chroma component (Cb, Cr) in the current picture as follows :</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>monochrome picture</td> <td>Desc</td> </tr> <tr> <td>01b</td> <td>4:2:0 picture</td> <td>Desc</td> </tr> <tr> <td>10b</td> <td>4:2:2 picture (not supported)</td> <td></td> </tr> <tr> <td>11b</td> <td>4:4:4 picture (not supported)</td> <td></td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>It is set to the value of the syntax element read from the current active SPS.The corresponding Monochrome Flag (monochrome_flag) can be derived from this field.</p>	Value	Name	Description	00b	monochrome picture	Desc	01b	4:2:0 picture	Desc	10b	4:2:2 picture (not supported)		11b	4:4:4 picture (not supported)	
Value	Name	Description															
00b	monochrome picture	Desc															
01b	4:2:0 picture	Desc															
10b	4:2:2 picture (not supported)																
11b	4:4:4 picture (not supported)																
9	Reserved	<p>Format: MBZ</p>															

MFX_AVC_IMG_STATE

	8	MbMvFormatFlag Use MB level MvFormat flag (Encoder Only)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>IGNORE</td> <td>HW PAK ignore MvFormat in the MB data. When bit 12 == 0, all MBs use packed MV format. When bit 12 == 1, each MB data must use unpacked MV format, 8MV when there is no minor MV involved, and 32MV if there are some minor MVs.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>FOLLOW</td> <td>HW PAK will follow MvFormat value set within each MB data.</td> </tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <p style="text-align: center; color: blue; font-weight: bold;">Programming Notes</p> They must take one of the two values: the 8MV unpacked format (MvFormat = 101b), and the 32MV unpacked format (MvFormat = 110b). This bit can be set only when MvUnpackedFlag (bit 12 of this register) is set otherwise system could hang. </div>	Value	Name	Description	0	IGNORE	HW PAK ignore MvFormat in the MB data. When bit 12 == 0, all MBs use packed MV format. When bit 12 == 1, each MB data must use unpacked MV format, 8MV when there is no minor MV involved, and 32MV if there are some minor MVs.	1	FOLLOW	HW PAK will follow MvFormat value set within each MB data.
Value	Name	Description										
0	IGNORE	HW PAK ignore MvFormat in the MB data. When bit 12 == 0, all MBs use packed MV format. When bit 12 == 1, each MB data must use unpacked MV format, 8MV when there is no minor MV involved, and 32MV if there are some minor MVs.										
1	FOLLOW	HW PAK will follow MvFormat value set within each MB data.										
	7	EntropyCodingFlag Entropy Coding Flag, entropy_coding_flag	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 55%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>CAVLC bit-serial encoding mode</td> <td>Desc</td> </tr> <tr> <td style="text-align: center;">1</td> <td>CABAC bit-serial encoding mode.</td> <td>Desc</td> </tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <p style="text-align: center; color: blue; font-weight: bold;">Programming Notes</p> It specifies one of the two possible bit stream encoding modes in the AVC. It is set to the value of the syntax element read from the current active PPS. </div>	Value	Name	Description	0	CAVLC bit-serial encoding mode	Desc	1	CABAC bit-serial encoding mode.	Desc
Value	Name	Description										
0	CAVLC bit-serial encoding mode	Desc										
1	CABAC bit-serial encoding mode.	Desc										
	6	ImgDisposableFlag Current Img Disposable Flag or Non-Reference Picture Flag	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>REFERENCE</td> <td>the current decoding picture may be used as a reference picture for others</td> </tr> <tr> <td style="text-align: center;">1</td> <td>DISPOSABLE</td> <td>the current decoding picture is not used as a reference picture (e.g. a B-picture cannot be a reference picture for any subsequent decoding)</td> </tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <p style="text-align: center; color: blue; font-weight: bold;">Programming Notes</p> It is derived from $\text{ImgDisposableFlag} = (\text{nal_ref_idc} == 0)$. nal_ref_idc is a syntax element from a NAL unit. When this flag is set, no reference picture and DMV are written out. This field is only valid for VLD decoding mode. </div>	Value	Name	Description	0	REFERENCE	the current decoding picture may be used as a reference picture for others	1	DISPOSABLE	the current decoding picture is not used as a reference picture (e.g. a B-picture cannot be a reference picture for any subsequent decoding)
Value	Name	Description										
0	REFERENCE	the current decoding picture may be used as a reference picture for others										
1	DISPOSABLE	the current decoding picture is not used as a reference picture (e.g. a B-picture cannot be a reference picture for any subsequent decoding)										
	5	ConstrainedIPredFlag Constrained Intra Prediction Flag, constrained_ipred_flag It is set to the value of the syntax element in the current active PPS.	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>INTRA_AND_INTER</td> <td>allows both intra and inter neighboring MB to be used in the intra-prediction encoding of the current MB.</td> </tr> </tbody> </table>	Value	Name	Description	0	INTRA_AND_INTER	allows both intra and inter neighboring MB to be used in the intra-prediction encoding of the current MB.			
Value	Name	Description										
0	INTRA_AND_INTER	allows both intra and inter neighboring MB to be used in the intra-prediction encoding of the current MB.										

MFX_AVC_IMG_STATE

	1	INTRA_ONLY	allows only to use neighboring Intra MBs in the intra-prediction encoding of the current MB. If the neighbor is an inter MB, it is considered as not available.									
4	<p>Direct8x8InfFlag Direct 8x8 Inference Flag, <code>direct_8x8_inference_flagIt</code> is set to the value of the syntax element in the current active SPS.It specifies the derivation process for luma motion vectors in the Direct MV coding modes (B_Skip, B_Direct_16x16 and B_Direct_8x8). When <code>frame_mbs_only_flag</code> is equal to 0, <code>direct_8x8_inference_flag</code> shall be equal to 1.It must be consistent with the <code>frame_mbs_only_flag</code> and <code>transform_8x8_mode_flag</code> settings.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>SUBBLOCK</td> <td>allows subpartitioning to go below 8x8 block size (i.e. 4x4, 8x4 or 4x8)</td> </tr> <tr> <td style="text-align: center;">1</td> <td>BLOCK</td> <td>allows processing only at 8x8 block size. MB Info is stored for 8x8 block size.</td> </tr> </tbody> </table>			Value	Name	Description	0	SUBBLOCK	allows subpartitioning to go below 8x8 block size (i.e. 4x4, 8x4 or 4x8)	1	BLOCK	allows processing only at 8x8 block size. MB Info is stored for 8x8 block size.
Value	Name	Description										
0	SUBBLOCK	allows subpartitioning to go below 8x8 block size (i.e. 4x4, 8x4 or 4x8)										
1	BLOCK	allows processing only at 8x8 block size. MB Info is stored for 8x8 block size.										
3	<p>Transform8x8Flag 8x8 IDCT Transform Mode Flag, <code>trans8x8_mode_flag</code> Specifies 8x8 IDCT transform may be used in this pictureIt is set to the value of the syntax element in the current active PPS.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>4x4</td> <td>no 8x8 IDCT Transform, only 4x4 IDCT transform blocks are present</td> </tr> <tr> <td style="text-align: center;">1</td> <td>8x8</td> <td>8x8 Transform is allowed</td> </tr> </tbody> </table>			Value	Name	Description	0	4x4	no 8x8 IDCT Transform, only 4x4 IDCT transform blocks are present	1	8x8	8x8 Transform is allowed
Value	Name	Description										
0	4x4	no 8x8 IDCT Transform, only 4x4 IDCT transform blocks are present										
1	8x8	8x8 Transform is allowed										
2	<p>FrameMbOnlyFlag Frame MB only flag, <code>frame_mbs_only_flagIt</code> is set to the value of the syntax element in the current active SPS.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>FALSE</td> <td>not true ; effectively enables the possibility of MBAFF mode.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>TRUE</td> <td>true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.</td> </tr> </tbody> </table>			Value	Name	Description	0	FALSE	not true ; effectively enables the possibility of MBAFF mode.	1	TRUE	true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.
Value	Name	Description										
0	FALSE	not true ; effectively enables the possibility of MBAFF mode.										
1	TRUE	true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.										
1	<p>MbaffFlameFlag MBAFF mode is active, <code>mbaff_frame_flag</code>.It is derived from <code>MbaffFrameFlag = (mb_adaptive_frame_field_flag && ! field_pic_flag)</code>. <code>mb_adaptive_frame_field_flag</code> is a syntax element in the current active SPS and <code>field_pic_flag</code> is a syntax element in the current Slice Header. They both are present only if <code>frame_mbs_only_flag</code> is 0. Although <code>mbaff_frame_flag</code> is a Slice Header parameter, its value is expected to be the same for all the slices of a picture.It must be consistent with the <code>mb_adaptive_frame_field_flag</code>, the <code>field_pic_flag</code> and the <code>frame_mbs_only_flag</code> settings.This bit is valid only when the <code>img_structure[1:0]</code> indicates the current picture is a frame.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>FALSE</td> <td>not in MBAFF mode</td> </tr> <tr> <td style="text-align: center;">1</td> <td>TRUE</td> <td>in MBAFF mode</td> </tr> </tbody> </table>			Value	Name	Description	0	FALSE	not in MBAFF mode	1	TRUE	in MBAFF mode
Value	Name	Description										
0	FALSE	not in MBAFF mode										
1	TRUE	in MBAFF mode										

MFX_AVC_IMG_STATE																									
	0	<p>FieldPicFlag Field picture flag, field_pic_flag, specifies the current slice is a coded field or not. It is set to the same value as the syntax element in the Slice Header. It must be consistent with the img_structure[1:0] and the frame_mbs_only_flag settings. Although field_pic_flag is a Slice Header parameter, its value is expected to be the same for all the slices of a picture.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">FRAME</td> <td>a slice of a coded frame</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">FIELD</td> <td>a slice of a coded field</td> </tr> </tbody> </table>	Value	Name	Description	0h	FRAME	a slice of a coded frame	1h	FIELD	a slice of a coded field														
Value	Name	Description																							
0h	FRAME	a slice of a coded frame																							
1h	FIELD	a slice of a coded field																							
5 [ExistsIf]Encode Only	31	<p>Trellis Quantization Enabled (TQEnb)</p> <table border="1"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>The TQ improves output video quality of AVC CABAC encoder by selecting quantized values for each non-zero coefficient so as to minimize the total R-D cost. This flag is only valid AVC CABAC mode. Otherwise, this flag should be disabled.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">Disable</td> <td>Use Normal</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">Enable</td> <td>Use Trellis quantization In VDENC mode, Enable Trellis Quantization in MFX. VDENC can control trellis Quantization based on it's programming.</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	Description	0h	Disable	Use Normal	1h	Enable	Use Trellis quantization In VDENC mode, Enable Trellis Quantization in MFX. VDENC can control trellis Quantization based on it's programming.												
	Format:	Enable																							
	Value	Name	Description																						
0h	Disable	Use Normal																							
1h	Enable	Use Trellis quantization In VDENC mode, Enable Trellis Quantization in MFX. VDENC can control trellis Quantization based on it's programming.																							
30:28	<p>Trellis Quantization Rounding (TQR) This rounding scheme is only applied to the quantized coefficients ranging from 0 to 1 when TQEnb is set to 1 in AVC CABAC mode. One of the following values is added to quantized coefficients before truncating fractional part.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">000b</td> <td></td> <td>Add 1/8</td> </tr> <tr> <td style="text-align: center;">001b</td> <td></td> <td>Add 2/8</td> </tr> <tr> <td style="text-align: center;">010b</td> <td></td> <td>Add 3/8</td> </tr> <tr> <td style="text-align: center;">011b</td> <td style="text-align: center;">[Default]</td> <td>Add 4/8 (rounding 0.5)</td> </tr> <tr> <td style="text-align: center;">100b</td> <td></td> <td>Add 5/8</td> </tr> <tr> <td style="text-align: center;">101b</td> <td></td> <td>Add 6/8</td> </tr> <tr> <td style="text-align: center;">110b</td> <td style="text-align: center;">Default</td> <td>Add 7/8 (Default rounding 0.875)</td> </tr> </tbody> </table>	Value	Name	Description	000b		Add 1/8	001b		Add 2/8	010b		Add 3/8	011b	[Default]	Add 4/8 (rounding 0.5)	100b		Add 5/8	101b		Add 6/8	110b	Default	Add 7/8 (Default rounding 0.875)
Value	Name	Description																							
000b		Add 1/8																							
001b		Add 2/8																							
010b		Add 3/8																							
011b	[Default]	Add 4/8 (rounding 0.5)																							
100b		Add 5/8																							
101b		Add 6/8																							
110b	Default	Add 7/8 (Default rounding 0.875)																							
	27	<p>Trellis Quantization Chroma Disable (TQChromaDisable) This signal is used to disable chroma TQ. To enable TQ for both luma and chroma, TQEnb=1, TQChromaDisable=0. To enable TQ only for luma, TQEnb=1, TQChromaDisable=1.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td></td> <td>Enable Trellis Quantization chroma</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">Default</td> <td>Disable Trellis Quantization chroma</td> </tr> </tbody> </table>	Value	Name	Description	0h		Enable Trellis Quantization chroma	1h	Default	Disable Trellis Quantization chroma														
Value	Name	Description																							
0h		Enable Trellis Quantization chroma																							
1h	Default	Disable Trellis Quantization chroma																							
	26:17	<p>Reserved</p> <table border="1"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																					
Format:	MBZ																								

MFX_AVC_IMG_STATE				
16	NonFirstPassFlag			
	This signals the current pass is not the first pass. It will imply designate HW behavior: e.g			
	Value	Name	Description	
	0h	Disable	Always use the MbQpY from initial PAK inline object for all passes of PAK	
	1h	Enable	Use MbQpY from stream-out buffer if MbRateCtrlFlag is set to 1	
	Reserved			
	Format:		MBZ	
	11:10	MinFrameWSizeUnits		
		This field is the Minimum Frame Size Units		
		Value	Name	Description
		00b	compatibility mode	Minimum Frame Size is in old mode (words, 2bytes)
		01b	16 byte	Minimum Frame Size is in 16bytes
		10b	4Kb	Minimum Frame Size is in 4Kbytes
	11b	16Kb	Minimum Frame Size is in 16Kbytes	
	9	MbRateCtrlFlag - MB level Rate Control Enabling Flag		
		MB Rate Control conformance mask		
In VDenc mode, this field must be zero as frame level rate control is used.				
Value		Name	Description	
0h		Disable	Apply accumulative delta QP for consecutive passes on top of the macroblock QP values in inline data	
1h	Enable	Apply RC QP delta to suggested QP values in Macroblock Status Buffer except the first pass.		
Programming Notes				
This field is ignored when MacroblockStatEnable is disabled or MB level Rate control flag for the current MB is disable in Macroblock Status Buffer.				
8	Reserved			
	Format:		MBZ	
7	Intra/InterMblpcmFlag - ForceIPCMControlMask			
	This field is to Force IPCM for Intra or Inter Macroblock size conformance mask.			
	Value	Name	Description	
	0h	Disable	Do not change intra or Inter macroblocks even	
	1h	Enable	Change intra or Inter macroblocks MB_type to IPCM	
Programming Notes				
This field is ignored when MacroblockStatEnable is disabled or MB level Intra MB conformance flag for the current MB is disable in Macroblock Status Buffer.				

MFX_AVC_IMG_STATE											
	6:4	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
	3	FrameSzUnderFlag - FrameBitRateMinReportMask This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	1h	Enable	set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit.
	Value	Name	Description								
	0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.								
1h	Enable	set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit.									
2	FrameSzOverFlag - FrameBitRateMaxReportMask This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax. <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>Set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	1	Enable	Set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.	
Value	Name	Description									
0	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.									
1	Enable	Set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.									
1	InterMbMaxBitFlag - InterMBMaxSizeReportMask This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize. <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable</td> <td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>Set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	1	Enable	Set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.	
Value	Name	Description									
0	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.									
1	Enable	Set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.									
0	IntraMbMaxBitFlag - IntraMBMaxSizeReportMask This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize. <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td> </tr> <tr> <td>1</td> <td>Enable</td> <td>set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	1	Enable	set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.	
Value	Name	Description									
0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.									
1	Enable	set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.									
6 [ExistsIf]Encode Only	31:28	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
27:16	InterMbMaxSz <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U12</td> </tr> </table> This field, Inter MB Conformance Max size limit,indicates the allowed max bit count size for Inter MB	Format:	U12								
Format:	U12										

MFX_AVC_IMG_STATE		
	15:12	Reserved Format: _____ MBZ
	11:0	IntraMbMaxSz Exists If: //Intra Only Format: U12 This field, Intra MB Conformance Max size limit, indicates the allowed max bit count size for Intra MB All IPCM MBs should ignore this Max size limit.
	7	Reserved Format: _____ MBZ
	8 [ExistsIf]Encode Only	SliceDeltaQpMax[3] Format: _____ S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count above FrameBitRateMax - first 1/8 region This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of (FrameBitRateMax, (FrameBitRateMax+FrameBitRateMaxDelta»3).
	23:16	SliceDeltaQpMax[2] Format: _____ U8 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/8 and below 1/4 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta»3), (FrameBitRateMax+FrameBitRateMaxDelta»2).
	15:8	SliceDeltaQpMax[1] Format: _____ S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/4 and below 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between and of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+FrameBitRateMaxDelta»2), (FrameBitRateMax+FrameBitRateMaxDelta»1).

MFX_AVC_IMG_STATE			
	<table border="1"> <tr> <td style="text-align: center;">7:0</td> <td> SliceDeltaQpPMax[0] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta, i.e., in the range of $((\text{FrameBitRateMax} + \text{FrameBitRateMaxDelta} \gg 1), \text{infinite})$. </td> </tr> </table>	7:0	SliceDeltaQpPMax[0] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta, i.e., in the range of $((\text{FrameBitRateMax} + \text{FrameBitRateMaxDelta} \gg 1), \text{infinite})$.
7:0	SliceDeltaQpPMax[0] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta, i.e., in the range of $((\text{FrameBitRateMax} + \text{FrameBitRateMaxDelta} \gg 1), \text{infinite})$.		
9 [ExistsIf]Encode Only	<table border="1"> <tr> <td style="text-align: center;">31:24</td> <td> SliceDeltaQpMin[3] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 region. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3), \text{FrameBitRateMin}]$. </td> </tr> </table>	31:24	SliceDeltaQpMin[3] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 region. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3), \text{FrameBitRateMin}]$.
	31:24	SliceDeltaQpMin[3] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 region. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3), \text{FrameBitRateMin}]$.	
	<table border="1"> <tr> <td style="text-align: center;">23:16</td> <td> SliceDeltaQpMin[2] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/8 and above 1/4. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3)]$. </td> </tr> </table>	23:16	SliceDeltaQpMin[2] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/8 and above 1/4. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3)]$.
	23:16	SliceDeltaQpMin[2] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/8 and above 1/4. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3)]$.	
<table border="1"> <tr> <td style="text-align: center;">15:8</td> <td> SliceDeltaQpMin[1] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/4 and above 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 1), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2)]$. </td> </tr> </table>	15:8	SliceDeltaQpMin[1] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/4 and above 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 1), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2)]$.	
15:8	SliceDeltaQpMin[1] Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/4 and above 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 1), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2)]$.		
<table border="1"> <tr> <td style="text-align: center;">7:0</td> <td> SliceDeltaQpMin[0] Format: S7 </td> </tr> </table>	7:0	SliceDeltaQpMin[0] Format: S7	
7:0	SliceDeltaQpMin[0] Format: S7		

MFX_AVC_IMG_STATE											
		<p>Range: [0:MAX_QP_DELTA]</p> <p>This field is the Slice Level Delta QP for bit-count below FrameBitRateMin - below 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bitcount for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta, i.e., in the range of [0, (FrameBitRateMin-FrameBitRateMinDelta»1).</p>									
10 [Exists] Encode Only	31	<p>FrameBitrateMaxUnit This field is the Frame Bitrate Maximum Limit Units.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Byte</td> <td>FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0</td> </tr> <tr> <td>1</td> <td>Kilo Byte</td> <td>FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0</td> </tr> </tbody> </table>	Value	Name	Description	0	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0	1	Kilo Byte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0
	Value	Name	Description								
	0	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0								
	1	Kilo Byte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0								
30	<p>FrameBitrateMaxUnitMode This field is the Frame Bitrate Maximum Limit Units.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>compatibility mode</td> <td>FrameBitRateMaxUnit is in old mode (128b/16Kb)</td> </tr> <tr> <td>1h</td> <td>New mode</td> <td>FrameBitRateMaxUnit is in new mode (32byte/4Kb)</td> </tr> </tbody> </table>	Value	Name	Description	0h	compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)	1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)	
Value	Name	Description									
0h	compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)									
1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)									
29:16	<p>FrameBitRateMax</p> <p>Format: U14</p> <p>This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. When FrameBitrateMaxUnitMode is 0 (compatibility mode) bits 16:27 should be used, bits 28 and 29 should be 0..</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[0-16383]</td> <td></td> <td>When FrameBitrateMaxUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.</td> </tr> <tr> <td>[0-16383]</td> <td></td> <td>When FrameBitrateMaxUnit = 1, this field is measured in unit of 4K bytes (1K=1024). The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.</td> </tr> </tbody> </table>	Value	Name	Description	[0-16383]		When FrameBitrateMaxUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.	[0-16383]		When FrameBitrateMaxUnit = 1, this field is measured in unit of 4K bytes (1K=1024). The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.	
Value	Name	Description									
[0-16383]		When FrameBitrateMaxUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.									
[0-16383]		When FrameBitrateMaxUnit = 1, this field is measured in unit of 4K bytes (1K=1024). The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.									
15	<p>FrameBitrateMinUnit This field is the Frame Bitrate Minimum Limit Units.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Byte</td> <td>FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0</td> </tr> <tr> <td>1</td> <td>Kilo</td> <td>FrameBitRateMax is in units of 4KBytes Bytes when</td> </tr> </tbody> </table>	Value	Name	Description	0	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0	1	Kilo	FrameBitRateMax is in units of 4KBytes Bytes when	
Value	Name	Description									
0	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0									
1	Kilo	FrameBitRateMax is in units of 4KBytes Bytes when									

MFX_AVC_IMG_STATE																	
		Byte	FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0														
	14	FrameBitrateMinUnitMode This field is the Frame Bitrate Minimum Limit Units. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>Compatibility mode</td> <td>FrameBitRateMaxUnit is in old mode (128b/16Kb)</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>New mode</td> <td>FrameBitRateMaxUnit is in new mode (32byte/4Kb)</td> </tr> </tbody> </table>		Value	Name	Description	0h	Compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)	1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)					
Value	Name	Description															
0h	Compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)															
1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)															
	13:0	FrameBitRateMin This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines minimum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count happen to be below this value. It takes on a value in the range of [0-16383]. When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0. WhenFrameBitrateMinUnit=0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit. WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.															
11 [ExistsIf]Encode Only	31	Slice Stats Streamout Enable Enable signal for Slice size streamout. When this bit is set, the slice size is written to SliceSize StreamOut Data (MFX_PIPE_BUF_ADDR_STATE DW 65,66,67)using one word. The slice size is in units of Bytes.															
	30:16	FrameBitRateMaxDelta Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px; height: 15px;"></td><td style="text-align: center;">U15</td></tr></table> This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode), only bits 16:27 should be used, bits 28, 29 and 30 should be 0. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0-32767]</td> <td></td> <td>WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.</td> </tr> <tr> <td style="text-align: center;">[0-32767]</td> <td></td> <td>WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.</td> </tr> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">[Default]</td> <td></td> </tr> </tbody> </table>			U15	Value	Name	Description	[0-32767]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.	[0-32767]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.	0h	[Default]	
	U15																
Value	Name	Description															
[0-32767]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.															
[0-32767]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.															
0h	[Default]																
	15	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px; height: 15px;"></td><td style="text-align: center;">MBZ</td></tr></table>			MBZ												
	MBZ																

MFX_AVC_IMG_STATE																																			
	<table border="1"> <tr> <td style="text-align: center;">14:0</td> <td> FrameBitRateMinDelta Range: [0-32767]. This field is used to select the slice delta QP when FrameBitRateMin is exceeded. It shares the same FrameBitRateMinUnit. When FrameBitRateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0. Note: HW requires the following condition $\text{FrameBitRateMinDelta} \leq 2 * \text{FrameBitRateMin}$ must be true, otherwise it may cause unpredictable behavior. When FrameBitRateMinUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024 KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit. When FrameBitRateMinUnit = 1, this field is measured in unit of 4K bytes (1K=1024). The frame rate in bytes is range from 0-128 MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit. </td> </tr> </table>	14:0	FrameBitRateMinDelta Range: [0-32767]. This field is used to select the slice delta QP when FrameBitRateMin is exceeded. It shares the same FrameBitRateMinUnit. When FrameBitRateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0. Note: HW requires the following condition $\text{FrameBitRateMinDelta} \leq 2 * \text{FrameBitRateMin}$ must be true, otherwise it may cause unpredictable behavior. When FrameBitRateMinUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024 KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit. When FrameBitRateMinUnit = 1, this field is measured in unit of 4K bytes (1K=1024). The frame rate in bytes is range from 0-128 MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.																																
14:0	FrameBitRateMinDelta Range: [0-32767]. This field is used to select the slice delta QP when FrameBitRateMin is exceeded. It shares the same FrameBitRateMinUnit. When FrameBitRateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12, 13 and 14 should be 0. Note: HW requires the following condition $\text{FrameBitRateMinDelta} \leq 2 * \text{FrameBitRateMin}$ must be true, otherwise it may cause unpredictable behavior. When FrameBitRateMinUnit = 0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024 KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit. When FrameBitRateMinUnit = 1, this field is measured in unit of 4K bytes (1K=1024). The frame rate in bytes is range from 0-128 MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.																																		
12	<table border="1"> <tr> <td style="text-align: center;">31:0</td> <td> Reserved Format: MBZ </td> </tr> </table>	31:0	Reserved Format: MBZ																																
31:0	Reserved Format: MBZ																																		
13	<table border="1"> <tr> <td style="text-align: center;">31:30</td> <td> Reserved Format: MBZ </td> </tr> <tr> <td style="text-align: center;">29</td> <td> Current Picture Has Performed MMCO5 Set to 1 if the current Pic has performed the memory_management_control_operation = 5. </td> </tr> <tr> <td style="text-align: center;">28:24</td> <td> <table border="1"> <tr> <td style="text-align: center;">Number of Reference Frames</td> <td>Format: U5</td> </tr> <tr> <td colspan="2">Range: Range 0 to MaxDpbSize (= 16 for Level 4.1)</td> </tr> <tr> <td colspan="2">Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.</td> </tr> </table> </td> </tr> <tr> <td style="text-align: center;">23:22</td> <td> <table border="1"> <tr> <td style="text-align: center;">Reserved</td> <td>Format: MBZ</td> </tr> </table> </td> </tr> <tr> <td style="text-align: center;">21:16</td> <td> <table border="1"> <tr> <td style="text-align: center;">Number of Active Reference Pictures from L1</td> <td>Format: U6-1</td> </tr> <tr> <td colspan="2">Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> </tr> <tr> <td style="text-align: center;">[0,31]</td> <td></td> </tr> </table> </td> </tr> <tr> <td style="text-align: center;">15:14</td> <td> <table border="1"> <tr> <td style="text-align: center;">Reserved</td> <td>Format: MBZ</td> </tr> </table> </td> </tr> <tr> <td style="text-align: center;">13:8</td> <td> <table border="1"> <tr> <td style="text-align: center;">Number of Active Reference Pictures from L0</td> <td>Format: U6-1</td> </tr> </table> </td> </tr> </table>	31:30	Reserved Format: MBZ	29	Current Picture Has Performed MMCO5 Set to 1 if the current Pic has performed the memory_management_control_operation = 5.	28:24	<table border="1"> <tr> <td style="text-align: center;">Number of Reference Frames</td> <td>Format: U5</td> </tr> <tr> <td colspan="2">Range: Range 0 to MaxDpbSize (= 16 for Level 4.1)</td> </tr> <tr> <td colspan="2">Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.</td> </tr> </table>	Number of Reference Frames	Format: U5	Range: Range 0 to MaxDpbSize (= 16 for Level 4.1)		Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.		23:22	<table border="1"> <tr> <td style="text-align: center;">Reserved</td> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ	21:16	<table border="1"> <tr> <td style="text-align: center;">Number of Active Reference Pictures from L1</td> <td>Format: U6-1</td> </tr> <tr> <td colspan="2">Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> </tr> <tr> <td style="text-align: center;">[0,31]</td> <td></td> </tr> </table>	Number of Active Reference Pictures from L1	Format: U6-1	Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture.		Value	Name	[0,31]		15:14	<table border="1"> <tr> <td style="text-align: center;">Reserved</td> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ	13:8	<table border="1"> <tr> <td style="text-align: center;">Number of Active Reference Pictures from L0</td> <td>Format: U6-1</td> </tr> </table>	Number of Active Reference Pictures from L0	Format: U6-1
31:30	Reserved Format: MBZ																																		
29	Current Picture Has Performed MMCO5 Set to 1 if the current Pic has performed the memory_management_control_operation = 5.																																		
28:24	<table border="1"> <tr> <td style="text-align: center;">Number of Reference Frames</td> <td>Format: U5</td> </tr> <tr> <td colspan="2">Range: Range 0 to MaxDpbSize (= 16 for Level 4.1)</td> </tr> <tr> <td colspan="2">Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.</td> </tr> </table>	Number of Reference Frames	Format: U5	Range: Range 0 to MaxDpbSize (= 16 for Level 4.1)		Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.																													
Number of Reference Frames	Format: U5																																		
Range: Range 0 to MaxDpbSize (= 16 for Level 4.1)																																			
Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.																																			
23:22	<table border="1"> <tr> <td style="text-align: center;">Reserved</td> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ																																
Reserved	Format: MBZ																																		
21:16	<table border="1"> <tr> <td style="text-align: center;">Number of Active Reference Pictures from L1</td> <td>Format: U6-1</td> </tr> <tr> <td colspan="2">Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> </tr> <tr> <td style="text-align: center;">[0,31]</td> <td></td> </tr> </table>	Number of Active Reference Pictures from L1	Format: U6-1	Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture.		Value	Name	[0,31]																											
Number of Active Reference Pictures from L1	Format: U6-1																																		
Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture.																																			
Value	Name																																		
[0,31]																																			
15:14	<table border="1"> <tr> <td style="text-align: center;">Reserved</td> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ																																
Reserved	Format: MBZ																																		
13:8	<table border="1"> <tr> <td style="text-align: center;">Number of Active Reference Pictures from L0</td> <td>Format: U6-1</td> </tr> </table>	Number of Active Reference Pictures from L0	Format: U6-1																																
Number of Active Reference Pictures from L0	Format: U6-1																																		

MFX_AVC_IMG_STATE								
		<p>Specifies the initial maximum reference index value minus 1 to access the L0 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L0 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Valid for both P and B pictures.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0,31]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,31]			
Value	Name							
[0,31]								
	7:0	<p>Initial QP Value</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%; text-align: center;">S7</td> </tr> <tr> <td colspan="2">Range: [-26,25]</td> </tr> <tr> <td colspan="2">Initial QP value for a Slice, extracted from PPS. It may further get modified by slice_qp_delta in slice header and mb_qp_delta in MB header.</td> </tr> </table>	Format:	S7	Range: [-26,25]		Initial QP value for a Slice, extracted from PPS. It may further get modified by slice_qp_delta in slice header and mb_qp_delta in MB header.	
Format:	S7							
Range: [-26,25]								
Initial QP value for a Slice, extracted from PPS. It may further get modified by slice_qp_delta in slice header and mb_qp_delta in MB header.								
14 [ExistsIf] Short Format only	31:24	<p>Log2_max_pic_order_cnt_lsb_minus4</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Short Format Only</td> </tr> </table> <p>It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent pic_order_cnt_lsb syntax element in the slice header. Unsigned</p>	Exists If:	//Short Format Only				
Exists If:	//Short Format Only							
	23:16	<p>Log2_max_frame_num_minus4</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Short Format Only</td> </tr> </table> <p>It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent frame_num syntax element in the slice header. Unsigned.</p>	Exists If:	//Short Format Only				
Exists If:	//Short Format Only							
	15	<p>deblocking_filter_control_present_flag</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Short Format Only</td> </tr> </table> <p>It is a PPS syntax element, indicates if more deblocking filter control syntax elements are present in the slice header.</p>	Exists If:	//Short Format Only				
Exists If:	//Short Format Only							
	14:12	<p>num_slice_groups_minus1</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Short Format Only</td> </tr> </table> <p>BitField It is a PPS syntax element. Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support. Desc</p>	Exists If:	//Short Format Only				
Exists If:	//Short Format Only							
	11	<p>redundant_pic_cnt_present_flag</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Short Format Only</td> </tr> </table> <p>It is a PPS syntax element. Use for Slice Header parsing only, to read-in redundant_pic_cnt, if any, but is not used by H/W, i.e. no support for redundant slice processing.</p>	Exists If:	//Short Format Only				
Exists If:	//Short Format Only							
	10:8	<p>slice_group_map_type</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Short Format Only</td> </tr> </table>	Exists If:	//Short Format Only				
Exists If:	//Short Format Only							

MFX_AVC_IMG_STATE													
		It is a PPS syntax element. Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.											
	7:4	Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	3:2	Pic_order_cnt_type <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Short Format Only</td> </tr> </table> <p>It is a SPS syntax element. Use for Slice Header parsing only.</p>	Exists If:	//Short Format Only									
Exists If:	//Short Format Only												
	1	Delta_pic_order_always_zero_flag <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Short Format Only</td> </tr> </table> <p>It is a SPS syntax element. Use for Slice Header parsing only.</p>	Exists If:	//Short Format Only									
Exists If:	//Short Format Only												
	0	Pic_order_present_flag <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Short Format Only</td> </tr> </table> <p>It is a PPS syntax element. Use for Slice Header parsing only.</p>	Exists If:	//Short Format Only									
Exists If:	//Short Format Only												
15 [ExistsIf] Short Format only	31:16	Curr Pic Frame Num <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Short Format Only</td> </tr> <tr> <td>Format:</td> <td>U16</td> </tr> </table> <p>Derived from Slice Header syntax element</p>	Exists If:	//Short Format Only	Format:	U16							
	Exists If:	//Short Format Only											
Format:	U16												
15:0	Slice Group Change Rate <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Short Format Only</td> </tr> <tr> <td>Format:</td> <td>U16-1</td> </tr> </table> <p>It is a PPS syntax element. Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.</p>	Exists If:	//Short Format Only	Format:	U16-1								
Exists If:	//Short Format Only												
Format:	U16-1												
16 [ExistsIf]: Short Format only	31	Inter View Order Disable <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Short Format Only</td> </tr> </table> <p>It indicates how to append inter-view picture into initial sorted reference list. (due to ambiguity in the MVC Spec)</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>Default [Default]</td> <td>View Order Ascending</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>Disable</td> <td>View ID Ascending</td> </tr> </tbody> </table>	Exists If:	//Short Format Only	Value	Name	Description	0h	Default [Default]	View Order Ascending	1h	Disable	View ID Ascending
		Exists If:	//Short Format Only										
		Value	Name	Description									
		0h	Default [Default]	View Order Ascending									
1h	Disable	View ID Ascending											
30:22	Reserved <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
21:18	Max View IDX1 <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>//Short Format Only</td> </tr> </table> <p>It is a PPS syntax element corresponding to Anchor/Non-Anchor Reference List1It</p>	Exists If:	//Short Format Only										
Exists If:	//Short Format Only												

MFX_AVC_IMG_STATE											
		indicates the maximum number of inter-view picture for Reference List L1									
	17:16	Reserved Format: _____ MBZ									
	15:12	Max View IDXLO Exists If: _____ //Short Format Only Reference ListL0It indicates the maximum number of inter-view picture for Reference List L0									
	11:10	Reserved Format: _____ MBZ									
	9:0	Current Frame View ID Exists If: _____ //Short Format Only It indicates the View ID of the current decoding frame									
17	31:22	Reserved Format: _____ MBZ									
	21:16	RhoDomain AverageMacroblockQP Exists If: _____ //[RhoDomain Rate Control] == 1 Format: _____ U6									
	15:9	Reserved Format: _____ MBZ									
	8	Extended RhoDomain Statistics Enable This parameter enables PAK to generate RhoDomain statistics from marcoblock QP and fractional QP computation. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: left;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Disable</td> <td>RhoDomain statistics generated from Frame Qp and no fractional QP computation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Enable</td> <td>Enable MB QP based RhoDomain statistics and fractional QP computation.</td> </tr> </tbody> </table> Note: By default, this is always enabled. Only the enable case will be validated.	Value	Name	Description	0	Disable	RhoDomain statistics generated from Frame Qp and no fractional QP computation.	1	Enable	Enable MB QP based RhoDomain statistics and fractional QP computation.
	Value	Name	Description								
	0	Disable	RhoDomain statistics generated from Frame Qp and no fractional QP computation.								
1	Enable	Enable MB QP based RhoDomain statistics and fractional QP computation.									
7:6	Reserved Format: _____ MBZ										
5:3	Fractional QP offset Format: _____ U3 Start position from the top of the Frame where increased Quantization parameter is added.										

MFX_AVC_IMG_STATE				
	2:0	Fractional QP input In a set of 8 rows, Qp is incremented by 1 for F_qp rows.		
18	31:0	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
19	31:0	Threshold Size in Bytes <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U32</td> </tr> </table> <p>When a slice exceeds this value in bytes, hardware will end current slice as soon as possible and insert a new slice boundary. Note there is no guarantee that the actual slice size will meet this value, it is a hint to the HW to end the slice as soon as possible (which could be 2-5 macroblocks in the future from this detection point).</p>	Format:	U32
Format:	U32			
20	31:0	Target Slice Size in Bytes <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U32</td> </tr> </table> <p>In VDENC mode, this field indicates the target slice size in Bytes for slice size conformance feature. When the actual slice size exceeds "Target slice size in Bytes", HW sets "VDENC Slice Overflow Error" bit in MFC_IMAGE_STATUS_CONTROL register.</p>	Format:	U32
Format:	U32			

MFX_AVC_REF_IDX_STATE

MFX_AVC_REF_IDX_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD mode); it is not need in decoder IT mode.</p> <p>The inline data of this command is interpreted differently for encoder as for decoder. For decoder, it is interpreted as RefIdx List L0/L1 as in AVC spec., and it matches with the AVC API data structure for decoder in VLD mode : RefPicList[2][32] (L0:L1, 0:31 RefPic). But for encoder, it is interpreted as a Reference Index Mapping Table for L0 and L1 reference pictures. For packing the bits at the output of PAK, the syntax elements must follow the definition of RefIdxL0/L1 list according to the AVC spec. However, the decoder pipeline was designed to use a variation of that standard definition, as such a conversion (mapping) is needed to support the hardware design. The Reference lists are needed in processing both P and B slice in AVC codec. For P-MB, only L0 list is used; for B-MB both L0 and L1 lists are needed. For a B-MB that is coded in L1-only Prediction, only L1 list is used.</p>			
Programming Notes			
<p>An application will create the RefPicList L0 and L1 and pass onto the driver. The content of each entry of RefPicList L0/L1[] is a 7-bit picture index. This picture index is the same as that of RefFrameList[] content. This picture index, however, is not defined the same as the frame store ID (0 to 16, 5-bits) we have implemented in H/W. Hence, driver is required to manage a table to convert between picture index and intel frame store ID. As such, the final RefPicList L0/L1[] that the driver passes onto the H/W is not the same as that defined.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_AVC_REF_IDX_STATE
		Format:	OpCode
	26:24	Command Opcode	
		Default Value:	1h AVC
		Format:	OpCode
	23:21	SubOpcodeA	
		Default Value:	0h MFX_AVC_REF_IDX_STATE
		Format:	OpCode
	20:16	SubOpcodeB	
		Default Value:	4h MFX_AVC_REF_IDX_STATE
Format:		OpCode	
15:12	Reserved		

MFX_AVC_REF_IDX_STATE											
		<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ										
	11:0	<p>DWord Length</p> <table border="1"> <tr> <td>Default Value:</td> <td>0008h</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>Excludes DWords 0,1</p>	Default Value:	0008h	Format:	=n					
Default Value:	0008h										
Format:	=n										
1	31:1	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ										
	0	<p>RefPicList Select</p> <p>Num_ref_idx_l1_active is resulted from the specifications in both PPS and Slice Header for the current slice. However, since the full reference list L0 and/or L1 are always sent, only present flags are specified instead.</p> <p>This parameter is specified for Intel interface only.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>RefPicList 0</td> <td>The list that followed represents RefList L0 (Decoder VLD mode) or Ref Idx Mapping Table L0 (Encoder PAK mode)</td> </tr> <tr> <td>1</td> <td>RefPicList1</td> <td>The list that followed represents RefList L1 (Decoder VLD mode) or Ref Idx Mapping Table L1 (Encoder PAK mode)</td> </tr> </tbody> </table>	Value	Name	Description	0	RefPicList 0	The list that followed represents RefList L0 (Decoder VLD mode) or Ref Idx Mapping Table L0 (Encoder PAK mode)	1	RefPicList1	The list that followed represents RefList L1 (Decoder VLD mode) or Ref Idx Mapping Table L1 (Encoder PAK mode)
Value	Name	Description									
0	RefPicList 0	The list that followed represents RefList L0 (Decoder VLD mode) or Ref Idx Mapping Table L0 (Encoder PAK mode)									
1	RefPicList1	The list that followed represents RefList L1 (Decoder VLD mode) or Ref Idx Mapping Table L1 (Encoder PAK mode)									
2..9	255:0	<p>Reference List Entry</p> <p>This set of fields is always present whenever this command is issued.</p> <p>It always specifies the full 32 reference pictures in the selected list, regardless they are "existing picture" or not. If a picture is non-existing, the corresponding entry should be set to all ones. Each list entry is 1 byte. A 32-bit DW can hold 4 list entries in the following format</p> <ul style="list-style-type: none"> • 31:24 entry X+3 (e.g. listY_3) • 23:16 entry X+2 (e.g. listY_2) • 15:8 entry X+1 (e.g. listY_1) • 7:0 entry X (e.g. listY_0) <p>X is replaced by the paddr[2:0] * 4 ; paddr[5:0] with 0x20 and 0x27, and Y is replaced by 0 or 1. The byte definition for a reference picture :</p> <ul style="list-style-type: none"> • Bit 7 : Non-Existing - indicates that frame store index that should have been at this entry did not exist and was replaced by an index 0 (a valid entry) for error concealment • Bit 6 : Long term bit - set this reference picture to be used 									
2..9		<p>Programming Notes:</p> <p>In VDENC mode, we support only three forward reference pictures. HW supports only 1:1 referece index to reference picture mapping.</p> <p>Reference index 0 should point to Reference picture 0, whose address is specified in MFX_PIPE_BUF_ADDR DW 19,20 (The reference picture numbers may be different from bit-stream reference picture)</p> <p>Reference index1 should point to Reference picture2, whose address is specified in MFX_PIPE_BUF_ADDR 23, 24 (The reference picture numbers may be different from bit-stream reference picture)</p> <p>Reference index2 should point to Reference picture4, whose address is specified in MFX_PIPE_BUF_ADDR 27,28 (The reference picture numbers may</p>									

MFX_AVC_REF_IDX_STATE	
<p>be different from bit-stream reference picture)</p>	<p>as long term reference</p> <ul style="list-style-type: none"> • Bit 5 : Field picture flag - indicates frame/field • Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation) <p>This is the final Reference List L0 or L1 after any reordering specified in the Slice Header as well as modified by the driver, and its indices values are all translated to the intel specification. If the reference picture is a frame (Bit5 = 1), frame store ID is always an even number. This list is used in outputting MV information by the BSD unit in VLD mode. DMV access also reads and writes Mvlist0 using this frame store ID. If this set of fields is interpreted as Reference Index Mapping Table L0/L1, the same field alignment is followed, i.e. 4 mapping entries per DW. Each mapping entry is one byte in size, but only the least significant 5 bits [4:0] is relevant. Driver should zero all the upper bits [7:5] for each entry.</p>

MFX_AVC_SLICE_STATE

MFX_AVC_SLICE_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).</p> <p>In VDEnc mode, this command is programmed for every super-slice. However not all parameters are allowed to change across super-slices.</p>			
Programming Notes			
<p>MFX_AVC_SLICE_STATE command is not issued for AVC Short Format Bitstream decode, instead MFD_AVC_SLICEADDR command is executed to retrieve the next slice MB Start Address X and Y by H/W itself.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_AVC_SLICE_STATE
		Format:	OpCode
	26:24	Command Opcode	
		Default Value:	1h AVC
		Format:	OpCode
	23:21	SubOpcodeA	
		Default Value:	0h MFX_AVC_SLICE_STATE
		Format:	OpCode
20:16	Command SubOpcodeB		
	Default Value:	3h MFX_AVC_SLICE_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length	Default Value:	8h DWORD_COUNT_n
		Format:	=n
		Excludes DWords 0,1	
1	31:4	Reserved	
		Format:	MBZ

MFX_AVC_SLICE_STATE												
	3:0	<p>Slice Type It is set to the value of the syntax element read from the Slice Header.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0000b</td> <td>P Slice</td> </tr> <tr> <td>0001b</td> <td>B Slice</td> </tr> <tr> <td>0010b</td> <td>I Slice</td> </tr> <tr> <td>0011b-1111b</td> <td>Reserved</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Bits[3:2] must be 0</p>	Value	Name	0000b	P Slice	0001b	B Slice	0010b	I Slice	0011b-1111b	Reserved
	Value	Name										
	0000b	P Slice										
	0001b	B Slice										
	0010b	I Slice										
	0011b-1111b	Reserved										
	2	31:30	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:		MBZ										
29:24		<p>Number of Reference Pictures in Inter-prediction List 1</p> <table border="1"> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>This field is valid only for encoding a B Slice, for which it is expected to have at least one entry in the reference list L1; otherwise (if Slice Type is not a B Slice), this field must be set to 0.This field can be derived for a B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L1 = NumRefIdxActiveMinus1[1] + 1.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0-32</td> <td></td> </tr> </tbody> </table>	Format:	U6	Value	Name	0-32					
Format:		U6										
Value		Name										
0-32												
23:22		<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:		MBZ										
21:16		<p>Number of Reference Pictures in Inter-prediction List 0</p> <table border="1"> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>This field is valid for encoding a P or B Slice, for which it is expected to have at least one entry in the reference list L0; otherwise (if Slice Type is not a P or B Slice), this field must be set to 0.This field can be derived for a P or B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L0 = NumRefIdxActiveMinus1[0] + 1.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0-32</td> <td></td> </tr> </tbody> </table>	Format:	U6	Value	Name	0-32					
Format:		U6										
Value	Name											
0-32												
15:11	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
10:8	<p>Log 2 Weight Denom Chroma</p> <table border="1"> <tr> <td>Format:</td> <td>U3</td> </tr> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0-7</td> <td></td> </tr> </tbody> </table>	Format:	U3	Value	Name	0-7						
Format:	U3											
Value	Name											
0-7												
7:3	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											

MFX_AVC_SLICE_STATE											
	2:0	Log 2 Weight Denom Luma Format: U3									
		It is the base 2 logarithm of the denominator for all Luma weighting factors. It is set to the value of the syntax element read from the Slice Header Pred_Weight_Table(). In VDEnc Mode, this value is programmed per super slice.									
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0-7</td> <td></td> </tr> </tbody> </table>	Value	Name	0-7						
		Value	Name								
0-7											
3	31:30	Weighted Prediction Indicator This field indicates the Weighted Prediction mode for a P or B Slice. It is a combined field corresponding to the syntax element WeightedBiPredIdc or WeightedPredFlag read from the current active PPS. <ul style="list-style-type: none"> If it is a B-Slice, these bits are interpreted as: <ul style="list-style-type: none"> 00b - Specifies the default weighted inter-prediction to be applied 01b - Specifies the explicit weighted inter-prediction to be applied 10b - Specifies the implicit weighted inter-prediction to be applied 11b - Reserved (not allowed) If it is a P Slice, these bits are interpreted as: <ul style="list-style-type: none"> 00b - Disables weighted inter-prediction (Default weighted) 01b - Enables weighted inter-prediction (Explicit weighted) 10b - 11b - Reserved 									
		Programming Notes									
		Only when in B Slice with Weighted_Pred_Idc = 1 (explicit weighted prediction), will there be a L1 and/or a L0 weight+offset tables being sent to the BSD unit through the Slice_State command. Only when in P Slice with Weighted_Pred_Idc = 1, will there be a L0 weight+offset table being sent to the BSD.									
		If Weighted_Pred_Idc != 1 for B Slice or Weighted_Pred_Idc = 0 for P Slice, no Slice_State command should be issued to send these tables. If still being issued, the data is read but ignored.									
		DXVA specifies Weighted_Bipred and Weighted_Pred in frame-level state. However, these two flags are combined and specified in slice level for both P and B slice type.									
29		Direct Prediction Type Type of direct prediction used for B Slices. This field is valid only for Slice_Type = B Slice; otherwise, it must be set to 0.									
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Temporal</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Spatial</td> </tr> </tbody> </table>	Value	Name	0	Temporal	1	Spatial			
		Value	Name								
0	Temporal										
1	Spatial										
28:27		Disable Deblocking Filter Indicator									
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td></td> <td>FilterInternalEdgesFlag is set equal to 1</td> </tr> <tr> <td style="text-align: center;">01b</td> <td></td> <td>Disable all deblocking operation, no deblocking parameter syntax element is</td> </tr> </tbody> </table>	Value	Name	Description	00b		FilterInternalEdgesFlag is set equal to 1	01b		Disable all deblocking operation, no deblocking parameter syntax element is
		Value	Name	Description							
00b		FilterInternalEdgesFlag is set equal to 1									
01b		Disable all deblocking operation, no deblocking parameter syntax element is									

MFX_AVC_SLICE_STATE			
			read; filterInternalEdgesFlag is set equal to 0
	10b		Macroblocks in different slices are considered not available; filterInternalEdgesFlag is set equal to 1
	11b	Reserved	Not defined in AVC
	26	Reserved	
		Format:	MBZ
	25:24	Cabac Init Idc[1:0] Specifies the index for determining the initialization table used in the context variable initialization process.	
		Value	Name
		0-2	
		Programming Notes	
		Cabac initialization is also dependent on the field/frame picture type, Slice type, and the current SliceQP value.	
	23:22	Reserved	
		Format:	MBZ
	21:16	Slice Quantization Parameter Quantization Parameter for current slice. Derived from PPS and slice_delta_qp syntax element in Slice Header. It is needed for CABAC context initialization and deblocking filter control. And it is also used as the starting QP value in the very first MB of a slice. It is in the range of unsigned integer 0 to 51, for 8-bit pixel bit-depth.	
	15:12	Reserved	
		Format:	MBZ
	11:8	Slice Beta Offset Div2	
		Format:	S3
		Range: [-6, 6] Inclusive	
		Specifies the offset used in accessing the deblocking filter strength tables.	
	7:4	Reserved	
		Format:	MBZ
	3:0	Slice Alpha C0 Offset Div2	
		Format:	S3
		Range: [-6, 6] Inclusive	
		Specifies the offset used in accessing the deblocking filter strength tables.	
4	31:24	Slice Vertical Position This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. The fields (Slice_MB_Start_Hor_Pos, Slice_MB_Start_Vert_Pos) are valid in VLD	

MFX_AVC_SLICE_STATE								
		<p>(decoding) mode only. They are ignored by hardware in decoding IT mode and encoding mode (whereas the position is provided by the per-macroblock object command).Derived</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.</td> </tr> </table>	Programming Notes		Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.			
Programming Notes								
Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.								
	23:16	<p>Slice Horizontal Position This field specifies the position in x-direction of the first macroblock in the Slice in unit of macroblocks. Derived</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.</td> </tr> </table>	Programming Notes		Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.			
Programming Notes								
Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.								
	15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
	14:0	<p>Slice Start Mb Num</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Exists If:</td> <td>//Decoder Only</td> </tr> </table> <p>The MB number (linear MB address in a picture) at the start of a Slice, it must match with the Slice Horizontal Position (Slice_MB_Start_Hor_Pos) and Vertical Position (Slice_MB_Start_Vert_Pos) in the picture.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">In creating the Phantom Slice for error concealment, this field should set to the total number of MB in the current picture + 1.</td> </tr> </table>	Exists If:	//Decoder Only	Programming Notes		In creating the Phantom Slice for error concealment, this field should set to the total number of MB in the current picture + 1.	
Exists If:	//Decoder Only							
Programming Notes								
In creating the Phantom Slice for error concealment, this field should set to the total number of MB in the current picture + 1.								
5	31:24	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	23:16	<p>Next Slice Vertical Position This field specifies the position in y-direction of the first macroblock in the next Slice in unit of macroblocks.This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to the height of picture (since y-direction is zero-based numbering).</p>						
	15:8	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
7:0	<p>Next Slice Horizontal Position This field specifies the position in x-direction of the first macroblock in the next Slice in unit of macroblocks.This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to 0.</p>							
6 Encoder Only	31	<p>Rate Control Counter Enable To enable the accumulation of bit allocation for rate controlThis field enables hardware Rate Control logic. The rest of the RC control fields are only valid when this field is set to 1. Otherwise, hardware ignores these fields.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Disable</td> </tr> </tbody> </table>	Value	Name	0	Disable		
		Value	Name					
0	Disable							

MFX_AVC_SLICE_STATE

	1	Enable
30	ResetRateControlCounter To reset the bit allocation accumulation counter to 0 to restart the rate control.	
	Value	Name
	0	Not Reset
	1	Reset
29:28	RC Triggler Mode	
	Value	Name
	00b	Always Rate Control Whereas RC becomes active if sum_act > sum_target or sum_act < sum_target
	01b	Gentle Rate Control whereas RC becomes active if sum_act > upper_midpt or sum_act < lower_midpt
	10b	Loose Rate Control whereas RC becomes active if sum_act > sum_max or sum_act < sum_min
	11b	Reserved
27:24	RC Stable Tolerance	
	Format:	U4
	This field specifies the tolerance required to deactivate RC once it has been triggered.	
	Value	Name
	0-15	
23	RC Panic Enable If this field is set to 1, RC enters panic mode when sum_act > sum_max. RC Panic Type field controls what type of panic behavior is invoked.	
	Value	Name
	0	Disable
	1	Enable
22	RC Panic Type This field selects between two RC Panic methods	
	Value	Name
	0	QP Panic
	1	CBP Panic
	Programming Notes	
	If it is set to 0, in panic mode, the macroblock QP is maxed out, setting to requested QP + QP_max_pos_mod. If it is set to 1, for an intra macroblock, AC CBPs are set to zero (note that DC CBPs are not modified). For inter macroblocks, AC and DC CBPs are forced to zero.	
21	MB Type Direct Conversion Disable	
	Exists If:	//B-Slice
	For all Macroblock type conversions in different slices, refer to Section "Macroblock Type	

MFX_AVC_SLICE_STATE		
	Conversion Rules" in the same volume.	
	Value	Name
	0	Enable direct mode conversion
	1	Disable direct mode conversion
	Programming Notes	
	This field is zero for all other slices other than B-Slice.	
20	MB Type Skip Conversion Disable	
	Exists If:	//P-Slice or B-Slice
	For all Macroblock type conversions in different slices, refer to Section "Macroblock Type Conversion Rules" in the same volume.	
	Value	Name
	0	Enable skip type conversion
	1	Disable skip type conversion
	Programming Notes	
	This field is zero for all other slices other than P_Slice or B-Slice. \	
19	Is Last Slice	
	It is used by the zero filling in the Minimum Frame Size test.	
	Set this only for the last slice group	
	Value	Name
	1	Current slice is the last slice of a picture
	0	Current slice is NOT the last slice of a picture
18	Reserved	
17	Header Insertion Present in Bitstream	
	Value	Name
	0	No header insertion into the output bitstream buffer, in front of the current slice encoded bits.
	1	Header insertion into the output bitstream buffer is present, and is in front of the current slice encoded bits.
	Programming Notes	
	Note: In VDEnc mode, the slice header PAK object maximum size is 25 DWs.	
	This need to be set only for super slice0.	
16	SliceData Insertion Present in Bitstream	
	Value	Name
	0	No Slice Data insertion into the output bitstream buffer

MFX_AVC_SLICE_STATE		
1		Slice Data insertion into the output bitstream buffer is present.
Programming Notes		
This bit should be set for all super-slices.		
15	Tail Insertion Present in bitstream	
Value	Name	Description
0		No tail insertion into the output bitstream buffer, after the current slice encoded bits
1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.
Programming Notes		
This bit should only be set for the last super slice. In VDENC mode, SW should insert 1000 VD_PIPELINE_FLUSH commands with VDENC_pipeline_Done set to 1 before inserting tail command. This is for delaying the tail insertion in HW. The HW recommendation is to insert tail only at the end of sequence to avoid performance loss since this restriction potentially cause performance degradation.		
14	Reserved	
	Format:	MBZ
13	EmulationByteSliceInsertEnable	
	To have PAK outputting SODB or EBSP to the output bitstream buffer	
Value	Name	Description
0		outputting RBSP
1		outputting EBSP
12	CabacZeroWordInsertionEnable	
	To pad the end of a SliceLayer RBSP to meet the encoded size requirement.	
Value	Name	Description
0		No Cabac_Zero_Word Insertion
1		Allow internal Cabac_Zero_Word generation and append to the end of RBSP(effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDS.
11:8	Reserved	
	Format:	MBZ
7:4	Slice ID [3:0]	
	To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.	
3:2	Reserved	
	Format:	MBZ

MFX_AVC_SLICE_STATE							
	<table border="1"> <tr> <td style="text-align: center;">1:0</td> <td> Stream ID [1:0] To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP. </td> </tr> </table>	1:0	Stream ID [1:0] To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.				
1:0	Stream ID [1:0] To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.						
7 Encoder Only	<table border="1"> <tr> <td style="text-align: center;">31:29</td> <td> Reserved Format: MBZ </td> </tr> </table>	31:29	Reserved Format: MBZ				
	31:29	Reserved Format: MBZ					
<table border="1"> <tr> <td style="text-align: center;">28:0</td> <td> Indirect PAK-BSE Data Start Address (Write) Exists If: //AVC Encode Mode This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0h,1FFFFFFh]</td> <td></td> </tr> </tbody> </table> </td> </tr> </table>	28:0	Indirect PAK-BSE Data Start Address (Write) Exists If: //AVC Encode Mode This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0h,1FFFFFFh]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0h,1FFFFFFh]		
28:0	Indirect PAK-BSE Data Start Address (Write) Exists If: //AVC Encode Mode This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0h,1FFFFFFh]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0h,1FFFFFFh]			
Value	Name						
[0h,1FFFFFFh]							
8 Encoder Only	<table border="1"> <tr> <td style="text-align: center;">31:24</td> <td> Magnitude of QP Max Negative Modifier Format: U8 This field specifies the lower limit of the QP modifier. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0-51</td> <td></td> </tr> </tbody> </table> </td> </tr> </table>	31:24	Magnitude of QP Max Negative Modifier Format: U8 This field specifies the lower limit of the QP modifier. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0-51</td> <td></td> </tr> </tbody> </table>	Value	Name	0-51	
	31:24	Magnitude of QP Max Negative Modifier Format: U8 This field specifies the lower limit of the QP modifier. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0-51</td> <td></td> </tr> </tbody> </table>	Value	Name	0-51		
	Value	Name					
	0-51						
	<table border="1"> <tr> <td style="text-align: center;">23:16</td> <td> Magnitude of QP Max Positive Modifier Format: U8 This field specifies the upper limit of the QP modifier. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table> </td> </tr> </table>	23:16	Magnitude of QP Max Positive Modifier Format: U8 This field specifies the upper limit of the QP modifier. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table>	Value	Name	0 - 15	
23:16	Magnitude of QP Max Positive Modifier Format: U8 This field specifies the upper limit of the QP modifier. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table>	Value	Name	0 - 15			
Value	Name						
0 - 15							
<table border="1"> <tr> <td style="text-align: center;">15:12</td> <td> Shrink Param - Shrink Resistance Format: U4 This field specifies the additional points added each time decreased correction is invoked. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table> </td> </tr> </table>	15:12	Shrink Param - Shrink Resistance Format: U4 This field specifies the additional points added each time decreased correction is invoked. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table>	Value	Name	0 - 15		
15:12	Shrink Param - Shrink Resistance Format: U4 This field specifies the additional points added each time decreased correction is invoked. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table>	Value	Name	0 - 15			
Value	Name						
0 - 15							
<table border="1"> <tr> <td style="text-align: center;">11:8</td> <td> Shrink Param - Shrink Init Format: U4 This field specifies the initial points required to trip decreased control. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table> </td> </tr> </table>	11:8	Shrink Param - Shrink Init Format: U4 This field specifies the initial points required to trip decreased control. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table>	Value	Name	0 - 15		
11:8	Shrink Param - Shrink Init Format: U4 This field specifies the initial points required to trip decreased control. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0 - 15</td> <td></td> </tr> </tbody> </table>	Value	Name	0 - 15			
Value	Name						
0 - 15							
<table border="1"> <tr> <td style="text-align: center;">7:4</td> <td> Grow Param - Grow Resistance Format: U4 This field specifies the additional points added each time increased correction is invoked. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;"></td> <td></td> </tr> </tbody> </table> </td> </tr> </table>	7:4	Grow Param - Grow Resistance Format: U4 This field specifies the additional points added each time increased correction is invoked. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;"></td> <td></td> </tr> </tbody> </table>	Value	Name			
7:4	Grow Param - Grow Resistance Format: U4 This field specifies the additional points added each time increased correction is invoked. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;"></td> <td></td> </tr> </tbody> </table>	Value	Name				
Value	Name						

MFX_AVC_SLICE_STATE																				
		0 - 15																		
	3:0	Grow Param - Grow Init Format: U4 This field specifies the initial points required to trip increased control. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0 - 15</td> <td></td> </tr> </tbody> </table>	Value	Name	0 - 15															
Value	Name																			
0 - 15																				
9 Encoder Only	31	RoundInterEnable Format: Enable When this bit is not set, RoundInter defaults to 2.																		
	30:28	RoundInter Format: U3 Rounding precision for Inter quantized coefficients <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%; text-align: center;">Value</th> <th style="width: 70%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>000b</td> <td>+1/16 [Default]</td> </tr> <tr> <td>001b</td> <td>+2/16</td> </tr> <tr> <td>010b</td> <td>+3/16</td> </tr> <tr> <td>011b</td> <td>+4/16</td> </tr> <tr> <td>100b</td> <td>+5/16</td> </tr> <tr> <td>101b</td> <td>+6/16</td> </tr> <tr> <td>110b</td> <td>+7/16</td> </tr> <tr> <td>111b</td> <td>+8/16</td> </tr> </tbody> </table>	Value	Name	000b	+1/16 [Default]	001b	+2/16	010b	+3/16	011b	+4/16	100b	+5/16	101b	+6/16	110b	+7/16	111b	+8/16
	Value	Name																		
	000b	+1/16 [Default]																		
	001b	+2/16																		
010b	+3/16																			
011b	+4/16																			
100b	+5/16																			
101b	+6/16																			
110b	+7/16																			
111b	+8/16																			
27	RoundIntraEnable Format: Enable When this bit is not set, RoundIntra defaults to 4.																			
26:24	RoundIntra Format: U3 Rounding precision for Intra quantized coefficients <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%; text-align: center;">Value</th> <th style="width: 70%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>000b</td> <td>+1/16 [Default]</td> </tr> <tr> <td>001b</td> <td>+2/16</td> </tr> <tr> <td>010b</td> <td>+3/16</td> </tr> <tr> <td>011b</td> <td>+4/16</td> </tr> <tr> <td>100b</td> <td>+5/16</td> </tr> <tr> <td>101b</td> <td>+6/16</td> </tr> <tr> <td>110b</td> <td>+7/16</td> </tr> </tbody> </table>	Value	Name	000b	+1/16 [Default]	001b	+2/16	010b	+3/16	011b	+4/16	100b	+5/16	101b	+6/16	110b	+7/16			
Value	Name																			
000b	+1/16 [Default]																			
001b	+2/16																			
010b	+3/16																			
011b	+4/16																			
100b	+5/16																			
101b	+6/16																			
110b	+7/16																			

MFX_AVC_SLICE_STATE		
		111b +8/16
10 Encoder Only	31:28	ClampValues - CV7
	27:24	CV6
	23:20	CV5
	23:20	Correct 6
	Format:	U4
	This field specifies the points used in the lowermost RC region when sum_act <= sum_min.	
	Value	Name
	0 - 15	
19:16	Correct 5	
Format:	U4	
This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.		
Value	Name	
	0 - 15	
15:12	Correct 4	
Format:	U4	
This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.		
Value	Name	
	0 - 15	
11:8	Correct 3	
Format:	U4	
This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt.		
Value	Name	
	0 - 15	
7:4	Correct 2	
Format:	U4	
This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max.		
Value	Name	
	0 - 15	
3:0	Correct 1	
Format:	U4	
This field specifies the points used in the topmost RC region when sum_act > sum_max.		
Value	Name	
	0 - 15	

MFX_AVC_SLICE_STATE

19:16	CV4																																																																																																																																																		
15:12	CV3																																																																																																																																																		
11:8	CV2																																																																																																																																																		
7:4	CV1																																																																																																																																																		
3:0	CV0 - Clamp Value 0																																																																																																																																																		
	Format:	U4																																																																																																																																																	
	<p>If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds $2CV0-1$, they are replaced with $2CV0-1$. For coefficients at locations marked as 'none', no clamping is performed. The following mappings are only applied to luma and chroma blocks/subblocks containing AC coefficients (blocks/subblocks with only DC coeffs will not be clamped).</p> <p>For 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <tr><td>none</td><td>CV7</td><td>CV5</td><td>CV4</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV4</td><td>CV3</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV4</td><td>CV3</td><td>CV1</td><td>CV0</td></tr> </table> <p>For 8x8 frame block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <tr><td>none</td><td>none</td><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td></tr> <tr><td>none</td><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td></tr> <tr><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td></tr> <tr><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td></tr> <tr><td>CV3</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td><td>CV0</td></tr> <tr><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV1</td><td>CV0</td><td>CV0</td><td>CV0</td></tr> </table> <p>For 4x4 field block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <tr><td>none</td><td>CV6</td><td>CV3</td><td>CV1</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV3</td><td>CV1</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV0</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV0</td></tr> </table> <p>For 8x8 field block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <tr><td>none</td><td>none</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td></tr> <tr><td>none</td><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV3</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV6</td><td>CV5</td><td>CV4</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV1</td><td>CV0</td></tr> <tr><td>CV6</td><td>CV5</td><td>CV4</td><td>CV3</td><td>CV2</td><td>CV2</td><td>CV1</td><td>CV0</td></tr> </table>			none	CV7	CV5	CV4	CV7	CV6	CV4	CV3	CV5	CV4	CV2	CV1	CV4	CV3	CV1	CV0	none	none	CV7	CV6	CV5	CV4	CV3	CV3	none	CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0	none	CV6	CV3	CV1	CV7	CV6	CV3	CV1	CV5	CV4	CV2	CV0	CV5	CV4	CV2	CV0	none	none	CV6	CV5	CV4	CV3	CV2	CV1	none	CV7	CV6	CV5	CV4	CV3	CV2	CV1	CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV1	CV7	CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV6	CV5	CV4	CV4	CV3	CV2	CV1	CV0	CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV0
none	CV7	CV5	CV4																																																																																																																																																
CV7	CV6	CV4	CV3																																																																																																																																																
CV5	CV4	CV2	CV1																																																																																																																																																
CV4	CV3	CV1	CV0																																																																																																																																																
none	none	CV7	CV6	CV5	CV4	CV3	CV3																																																																																																																																												
none	CV7	CV6	CV5	CV4	CV3	CV3	CV2																																																																																																																																												
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2																																																																																																																																												
CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1																																																																																																																																												
CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1																																																																																																																																												
CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0																																																																																																																																												
CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0																																																																																																																																												
CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0																																																																																																																																												
none	CV6	CV3	CV1																																																																																																																																																
CV7	CV6	CV3	CV1																																																																																																																																																
CV5	CV4	CV2	CV0																																																																																																																																																
CV5	CV4	CV2	CV0																																																																																																																																																
none	none	CV6	CV5	CV4	CV3	CV2	CV1																																																																																																																																												
none	CV7	CV6	CV5	CV4	CV3	CV2	CV1																																																																																																																																												
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV1																																																																																																																																												
CV7	CV6	CV5	CV4	CV3	CV2	CV2	CV1																																																																																																																																												
CV6	CV5	CV4	CV4	CV3	CV2	CV1	CV0																																																																																																																																												
CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV0																																																																																																																																												

MFX_AVC_SLICE_STATE											
		CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0		
		CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0		
		Value					Name				
		0 - 15									

MFX_AVC_WEIGHTOFFSET_STATE

MFX_AVC_WEIGHTOFFSET_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes). However, since for AVC decoder VLD and IT modes, and AVC encoder mode, the implicit weights are computed in hardware, this command is not issued. For encoder, regardless of the type of weight calculation is active for the current slice (default, implicit or explicit), they are all sent to the PAK as if they were all in explicit mode. However, for implicit weight and offset, each entry contains only a 16-bit weight and no offset (offset = 0 always in implicit mode and can be hard-coded inside the hardware).The weights (and offsets) are needed in processing both P and B slice in AVC codec. For P-MB, at most only L0 list is used; for B-MB both L0 and L1 lists may be needed. For a B-MB that is coded in L1-only Prediction, only L1 list is sent.The content of this command matches with the AVC API data structure for explicit prediction mode only : Weights[2][32][3][2] (L0:L1, 0:31 RefPic, Y:Cb:Cr, W:0)</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_AVC_WEIGHTOFFSET_STATE
	26:24	Media Command Opcode	
		Default Value:	1h AVC_COMMON
	23:21	SubOpcode A	
Default Value:		0h	
20:16	SubOpcode B		
	Default Value:	5h	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	60h Excludes DWord (0,1)	
	Format:	=n	
1	31:1	Reserved	
		Format:	MBZ

MFX_AVC_WEIGHTOFFSET_STATE

0	<p>Weight and Offset Select</p> <p>It must be set in consistent with the WeightedPredFlag and WeightedBiPredIdc in the Img_State command. This parameter is specified for Intel interface only. For implicit even though only one entry may be used, still loading the whole 32-entry table.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Weight and Offset L0 table</td> <td>The list that followed is associated with the weight and offset for RefPicList L0</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Weight and Offset L1 table</td> <td>The list that followed is associated with the weight and offset for RefPicList L1</td> </tr> </tbody> </table>	Value	Name	Description	0	Weight and Offset L0 table	The list that followed is associated with the weight and offset for RefPicList L0	1	Weight and Offset L1 table	The list that followed is associated with the weight and offset for RefPicList L1
Value	Name	Description								
0	Weight and Offset L0 table	The list that followed is associated with the weight and offset for RefPicList L0								
1	Weight and Offset L1 table	The list that followed is associated with the weight and offset for RefPicList L1								
2..97	<p>3071:0 WeightOffset</p> <p>WeightOffset[L=L0=0 or L1=1][i=0 to 31][Y=0/Cb=1/Cr=2][weight=0/offset=1]WeightOffset[L][i=0][Y=0][Weight=0], WeightOffset[L][i=0][Y=0][Offset=1]WeightOffset[L][i=0][Cb=1][Weight=0], WeightOffset[L][i=0][Cb=1][Offset=1]WeightOffset[L][i=0][Cr=2][Weight=0], WeightOffset[L][i=0][Cr=2][Offset=1]:WeightOffset[L][i=31][Y=0][Weight=0], WeightOffset[L][i=31][Y=0][Offset=1]WeightOffset[L][i=31][Cb=1][Weight=0], WeightOffset[L][i=31][Cb=1][Offset=1]WeightOffset[L][i=31][Cr=2][Weight=0], WeightOffset[L][i=31][Cr=2][Offset=1]</p> <p>Format for explicit: Both Weight and Offset are S15 in two's compliment, with a valid range from -128 to 128 Format for implicit: S15</p> <p>This set of fields is always present whenever this command is issued. The full table, one entry for each reference picture, is always specified. Any reference list L0/L1[i] that does not exist, the corresponding weight and offset are set to 0. Weight and Offset are 2 byte each. A pair of Weight and Offset forms a dword, with Weight in the LOWER word and Offset in the HIGHER word. WeightOffset[L0=0][i=0 to 31][Y=0] (i.e. luma_weight_I0[i]) are specified for the weighting and offset factors applied to the luma prediction value for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When luma_weight_I0_flag (Slice Header syntax element) is equal to 1, the value of luma_weight_I0[i] shall be in the range of -128 to 127. When luma_weight_I0_flag is equal to 0, luma_weight_I0[i] shall be inferred to be equal to 2luma_log2_weight_denom for RefPicList0[i]. luma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cb=1] (i.e. chromaCb_weight_I0[i]) are specified for the weighting and offset factors applied to the chroma Cb prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_I0_flag (Slice Header syntax element) is equal to 1, the value of chromaCb_weight_I0[i] shall be in the range of -128 to 127. When chroma_weight_I0_flag is equal to 0, chromaCb_weight_I0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i]. chroma_log2_weight_denom is a Slice Header syntax element. WeightOffset[L0=0][i=0 to 31][Cr=2] (i.e. chromaCr_weight_I0[i]) are specified for the weighting and offset factors applied to the chroma Cr prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_I0_flag (Slice Header syntax element) is equal to 1, the value of chromaCr_weight_I0[i] shall be in the range of -128 to 127. When chroma_weight_I0_flag is equal to 0, chromaCr_weight_I0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i].</p>									

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This frame-level state command is used to specify all the buffer base addresses needed for the operation of the AVC Bit Stream Processing Units (for decoder, it is BSD Unit; for encoder, it is BSE Unit)For both encoder and decoder, currently it is assumed that all codec standards can share the same BSP_BUF_BASE_STATE. The simplicity of this command is the result of moving all the direct MV related processing into the ENC Subsystem. Since all implicit weight calculations and directMV calculations are done in ENC and all picture buffer management are done in the Host, there is no need to provide POC (POC List - FieldOrderCntList, CurrPic POC - CurrFieldOrderCnt) information to PAK. For decoder, all the direct mode information are sent in a separate slice-level command (AVC_DIRECTMODE_STATE command).In addition, in Encoder, the row stores for CABAC encoding and MB Parameters Construction (MPC) are combined into one single row store.The row stores specified in this command do not combine with those specified in the MFC_PIPE_BUF_ADDR_STATE command for hardware simplification reason.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Pipeline
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	4h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	8h Excludes DWord (0,1)	
	Format:	=n	
1	31:6	BSD/MPC Row Store Scratch Buffer Base Address - Read/Write	
		This field provides the base address of the scratch buffer used by BSD (decoder) andMPC	

MFX_BSP_BUF_BASE_ADDR_STATE

		<p>(encoder) unit to store MB information of the previous row for coding each macroblock in the current row. It is a private buffer used by the BSD (decoder) and MPC (encoder) hardware only. Its content is not accessible by software. This Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address this Row Store.</p> <p>For AVC BSD, 2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MB for non-MBAFF. So, to support 256 MBs per row (4K screen resolution), $2 * 256 * 64$ bytes = 32,768 bytes are required. Cacheline alignment should be followed. For AVC MPC, 1 cacheline for non-MBAFF, 2 cachelines for MBAFF per MB. For VC1, the BSD row store is 512-bit (one cacheline) per MB, times the number of MBs per picture MB row.</p>																		
		Programming Notes																		
		<p>This is one of the four RowStore Scratch Buffers which can be programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cacheline address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage.</p> <p><i>(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).</i></p>																		
	5:0	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>		Format:	MBZ															
Format:	MBZ																			
2	31:16	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>		Format:	MBZ															
Format:	MBZ																			
	15:0	BSD/MPC Row Store Scratch Buffer Base Address - Read/Write [47:32] <p>This field is for the upper range of BSD/MPC Row Store Scratch Buffer Base Address.</p> <p>This field is used for 48-bit addressing.</p>																		
3	31:15	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>		Format:	MBZ															
Format:	MBZ																			
	14:13	BSD/MPC Row Store Scratch Buffer - Tiled Resource Mode <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U2</td> </tr> </table> <p>For Media Surfaces: This field specifies the tiled resource mode.</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>		Format:	U2	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
Format:	U2																			
Value	Name	Description																		
0h	TRMODE_NONE	No tiled resource																		
1h	TRMODE_TILEYF	4KB tiled resources																		
2h	TRMODE_TILEYS	64KB tiled resources																		
3h	Reserved																			
	12	BSD/MPC Row Store Scratch Buffer Cache Select <p>This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC.</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table>		Value	Name	Description														
Value	Name	Description																		

MFX_BSP_BUF_BASE_ADDR_STATE		
	0	Buffer going to LLC
	1	Buffer going to Internal Media Storage
11:9	Reserved	
	Format:	MBZ
8:7	BSD/MPC Row Store Scratch Buffer - Arbitration Priority Control	
	Format:	U2
	This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.	
	Value	Name
	00b	Highest priority
	01b	Second highest priority
	10b	Third highest priority
	11b	Lowest priority
6:1	BSD/MPC Row Store Scratch Buffer - Index to Memory Object Control State (MOCS) Tables	
	Format:	U6
	The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.	
0	Reserved	
4	31:6	MPR Row Store Scratch Buffer Base Address - Read/Write (Decoder Only) This field provides the base address of the scratch buffer used by decoder's MPR unit to store MB information of the previous row for decoding each macroblock in the current row. It is a private buffer used by the MPR hardware only. Its content is not accessible by software.
	Programming Notes	
	The MPR Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of each macroblock to address the MPR Row Store. Except ILDB Control Data, all other operations does not cross slice boundary. This field is specified in frame-level.2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MB for non-MBAFF, So, to support 256 MBs per row (4K screen resolution), 2 * 256 * 64 bytes = 32,768 bytes are required. Cacheline alignment should be followed. This field is only valid for AVC decoder mode	
	This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be cache inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage <i>(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).</i>	
	5:0	Reserved
	Format:	MBZ

MFx_BSP_BUF_BASE_ADDR_STATE																	
5	31:16	Reserved Format: _____ MBZ															
	15:0	MPR Row Store Scratch Buffer Base Address - Read/Write [47:32] This field is for the upper range of MPR Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.															
6	31:15	Reserved Format: _____ MBZ															
	14:13	MPR Row Store Scratch Buffer - Tiled Resource Mode Format: _____ U2 For Media Surfaces: This field specifies the tiled resource mode. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
	1h	TRMODE_TILEYF	4KB tiled resources														
	2h	TRMODE_TILEYS	64KB tiled resources														
	3h	Reserved															
12	MPR Row Store Scratch Buffer Cache Select This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>Buffer going to LLC</td> </tr> <tr> <td>1</td> <td></td> <td>Buffer going to Internal Media Storage</td> </tr> </tbody> </table>	Value	Name	Description	0		Buffer going to LLC	1		Buffer going to Internal Media Storage							
Value	Name	Description															
0		Buffer going to LLC															
1		Buffer going to Internal Media Storage															
11:9	Reserved Format: _____ MBZ																
8:7	MPR Row Store Scratch Buffer - Arbitration Priority Control Format: _____ U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">Value</th> <th style="width: 75%;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority						
Value	Name																
00b	Highest priority																
01b	Second highest priority																
10b	Third highest priority																
11b	Lowest priority																
6:1	MPR Row Store Scratch Buffer - Index to Memory Object Control State (MOCS) Tables Format: _____ U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.																
0	Reserved																

MFx_BSP_BUF_BASE_ADDR_STATE																			
7	31:6	<p>Bitplane Read Buffer Base Address</p> <p>It must be cacheline aligned (i.e. 64 bytes address boundary), so lower bit 0 to 5 are used for controlling information.(In Cantiga, this field must be dword aligned.)Bitplane buffer is a linear buffer. In VC1 Long format, it is written by an application. In VC1 Short Format, it is written and read by H/W only.For VC1 intel Long Format : it is a read-only bufferFor VC1 DXVA2 Short Format : it is a write and a read bufferThis field is only valid for VC1 decoder mode.</p>																	
	5:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ															
Format:	MBZ																		
8	31:16	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ															
	Format:	MBZ																	
15:0	<p>Bitplane Read Buffer Base Address - Read/Write [47:32]</p> <p>This field is for the upper range of Bitplane Read Buffer Base Address.This field is used for 48-bit addressing.</p>																		
9	31:15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ															
	Format:	MBZ																	
	14:13	<p>Bitplane Read Buffer - Tiled Resource Mode</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>For Media Surfaces:This field specifies the tiled resource mode.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Format:	U2	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Format:	U2																	
	Value	Name	Description																
	0h	TRMODE_NONE	No tiled resource																
1h	TRMODE_TILEYF	4KB tiled resources																	
2h	TRMODE_TILEYS	64KB tiled resources																	
3h	Reserved																		
12:9	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ																
Format:	MBZ																		
8:7	<p>Bitplane Read Buffer - Arbitration Priority Control</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Format:	U2	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority						
Format:	U2																		
Value	Name																		
00b	Highest priority																		
01b	Second highest priority																		
10b	Third highest priority																		
11b	Lowest priority																		
6:1	<p>Bitplane Read Buffer - Index to Memory Object Control State (MOCS) Tables</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers.The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.</p>	Format:	U6																
Format:	U6																		

MFX_BSP_BUF_BASE_ADDR_STATE		
	0	Reserved

MFX_DBK_OBJECT

MFX_DBK_OBJECT			
Source:		VideoCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_DBK_OBJECT
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	0h Common
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	9h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0Bh Excludes DWord (0,1)	
	Format:	=n	
	Note: Regardless of the mode, inline data must be present in this command		
1	31:6	Pre Deblocking Source Address	
		Format:	GraphicsAddress[31:6]
	Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit).		
5:0	Reserved		
	Format:	MBZ	
2	31:16	Reserved	
		Format:	MBZ

MFX_DBK_OBJECT																	
3	15:0	Pre Deblocking Source Address High Format: GraphicsAddress[47:32] This field is for the upper range of Pre-Deblocking Source Address. This field is used for 48-bit addressing.															
	31:15	Reserved Format: MBZ															
	14:13	Pre Deblocking Source - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td style="text-align: center;">2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td style="text-align: center;">3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
	1h	TRMODE_TILEYF	4KB tiled resources														
	2h	TRMODE_TILEYS	64KB tiled resources														
	3h	Reserved															
	12:11	Reserved Format: MBZ															
	10	Pre Deblocking Source - Memory Compression Mode Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter, Media Memory Compression section for more details. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Vertical Compression Mode</td> </tr> </tbody> </table>	Value	Name	1	Vertical Compression Mode											
Value	Name																
1	Vertical Compression Mode																
9	Pre Deblocking Source - Memory Compression Enable Format: Enable Memory compression will be attempted for this surface. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Compression Disable</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Compression Enable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable	1	Compression Enable										
Value	Name																
0	Compression Disable																
1	Compression Enable																
8:7	Pre Deblocking Source - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td>Highest priority</td> </tr> <tr> <td style="text-align: center;">01b</td> <td>Second highest priority</td> </tr> <tr> <td style="text-align: center;">10b</td> <td>Third highest priority</td> </tr> <tr> <td style="text-align: center;">11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority						
Value	Name																
00b	Highest priority																
01b	Second highest priority																
10b	Third highest priority																
11b	Lowest priority																
6:1	Pre Deblocking Source - Index to Memory Object Control State (MOCS) Tables The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.																

MFX_DBK_OBJECT																	
	0	Reserved															
4	31:6	Deblocking Control Address Format: <table border="1" style="display: inline-table;"><tr><td>GraphicsAddress[31:6]</td></tr></table> Specifies the 4K byte aligned frame buffer address as input MB-level deblocking parameters to control the way hardware deblock the each micro-block. One 512-bit cacheline is allocated for each Macroblock in raster scan order.	GraphicsAddress[31:6]														
	GraphicsAddress[31:6]																
5:0	Reserved Format: <table border="1" style="display: inline-table;"><tr><td>MBZ</td></tr></table>	MBZ															
MBZ																	
5	31:16	Reserved Format: <table border="1" style="display: inline-table;"><tr><td>MBZ</td></tr></table>	MBZ														
	MBZ																
15:0	Deblocking Control Address High Format: <table border="1" style="display: inline-table;"><tr><td>GraphicsAddress[47:32]</td></tr></table> This field is for the upper range of Deblocking Control Address (DeblockCntrlAddr). This field is used for 48-bit addressing.	GraphicsAddress[47:32]															
GraphicsAddress[47:32]																	
6	31:15	Reserved Format: <table border="1" style="display: inline-table;"><tr><td>MBZ</td></tr></table>	MBZ														
	MBZ																
	14:13	Deblocking Control - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td style="text-align: center;">2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td style="text-align: center;">3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
1h	TRMODE_TILEYF	4KB tiled resources															
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
12:11	Reserved Format: <table border="1" style="display: inline-table;"><tr><td>MBZ</td></tr></table>	MBZ															
MBZ																	
10	Deblocking Control - Memory Compression Mode Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter, Media Memory Compression section for more details. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Horizontal Compression Mode</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Vertical Compression Mode</td> </tr> </tbody> </table>	Value	Name	0	Horizontal Compression Mode	1	Vertical Compression Mode										
Value	Name																
0	Horizontal Compression Mode																
1	Vertical Compression Mode																
9	Deblocking Control - Memory Compression Enable Format: <table border="1" style="display: inline-table;"><tr><td>Enable</td></tr></table> Memory compression will be attempted for this surface. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Compression Disable</td> </tr> </tbody> </table>	Enable	Value	Name	0	Compression Disable											
Enable																	
Value	Name																
0	Compression Disable																

MFX_DBK_OBJECT																	
	8:7	<p>Deblocking Control - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority					
	Value	Name															
	00b	Highest priority															
	01b	Second highest priority															
	10b	Third highest priority															
11b	Lowest priority																
6:1	<p>Deblocking Source - Index to Memory Object Control State (MOCS) Tables The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.</p>																
0	Reserved																
7	31:6	<p>Deblocking Destination Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:6]</td> </tr> </table> <p>Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)</p>	Format:	GraphicsAddress[31:6]													
	Format:	GraphicsAddress[31:6]															
5:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
8	31:16	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
15:0	<p>Deblocking Destination Address High</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> <p>This field is for the upper range of Deblocking Destination Address. This field is used for 48-bit addressing.</p>	Format:	GraphicsAddress[47:32]														
Format:	GraphicsAddress[47:32]																
9	31:15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
	14:13	<p>Deblocking Destination - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
1h	TRMODE_TILEYF	4KB tiled resources															
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
12:11	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
10	Deblocking Destination - Memory Compression Mode																

MFX_DBK_OBJECT												
		<p>Distinguishes Vertical from Horizontal compression. Please refer to vol1aMemory Data Formats chapter, Media Memory Compression section for more details.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Horizontal Compression Mode</td> </tr> </tbody> </table>	Value	Name	0	Horizontal Compression Mode						
Value	Name											
0	Horizontal Compression Mode											
	9	<p>Deblocking Destination - Memory Compression Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Memory compression will be attempted for this surface.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Compression Disable</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	0	Compression Disable				
Format:	Enable											
Value	Name											
0	Compression Disable											
	8:7	<p>Deblocking Destination - Arbitration Priority Control</p> <p>This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td>Highest priority</td> </tr> <tr> <td style="text-align: center;">01b</td> <td>Second highest priority</td> </tr> <tr> <td style="text-align: center;">10b</td> <td>Third highest priority</td> </tr> <tr> <td style="text-align: center;">11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority
Value	Name											
00b	Highest priority											
01b	Second highest priority											
10b	Third highest priority											
11b	Lowest priority											
	6:1	<p>Deblocking Destination - Index to Memory Object Control State (MOCS) Tables</p> <p>The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.</p>										
	0	Reserved										
10	31:6	<p>Deblock Row Store Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:6]</td> </tr> </table> <p>This field provides the base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store.</p>	Format:	GraphicsAddress[31:6]								
Format:	GraphicsAddress[31:6]											
	5:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
11	31:16	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	15:0	<p>Deblock Row Store Address High</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> <p>This field is for the upper range of Deblock Row Store Address (DeblockRowStoreAddr). This field is used for 48-bit addressing.</p>	Format:	GraphicsAddress[47:32]								
Format:	GraphicsAddress[47:32]											
12	31:15	Reserved										

MFX_DBK_OBJECT

MFX_DBK_OBJECT		
	Format:	MBZ
14:13	Deblock Row Store - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode.	
	Value	Name
	0h	TRMODE_NONE
	1h	TRMODE_TILEYF
	2h	TRMODE_TILEYS
	3h	Reserved
12:11	Reserved	
	Format:	MBZ
10	Deblock Row Store - Memory Compression Mode Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter, Media Memory Compression section for more details.	
	Value	Name
	0	Horizontal Compression Mode
9	Deblock Row Store - Memory Compression Enable Format: Enable	
	Memory compression will be attempted for this surface.	
	Value	Name
	0	Compression Disable
8:7	Deblock Row Store - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.	
	Value	Name
	00b	Highest priority
	01b	Second highest priority
	10b	Third highest priority
	11b	Lowest priority
6:1	CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.	
0	Reserved	

MFX_FQM_STATE

MFX_FQM_STATE				
Source:	VideoCS			
Length Bias:	2			
<p>This is a common state command for AVC encoder modes. For encoder, it represents both the forward QM matrices as well as the decoding QM matrices. This is a Frame-level state. Only Scaling Lists specified by an application are being sent to the hardware. The driver is responsible for determining the final set of scaling lists to be used for decoding the current slice, based on the AVC Spec Table 7-2 (Fall-Back Rules A and B). In MFX AVC PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order. But the Forward Q scaling lists are sent in column-wise raster order (column-by-column) to simplify the H/W. Driver will perform all the scan order conversion for both ForwardQ and IQ.</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	28:27	Pipeline	
			Default Value:	2h MFX_MULTI_DW
			Format:	OpCode
	26:24	26:24	Media Command Opcode	
			Default Value:	0h MFX_COMMON_STATE
			Format:	OpCode
	23:21	23:21	SubOpcode A	
Default Value:			0h	
Format:			OpCode	
20:16	20:16	SubOpcode B		
		Default Value:	8h	
		Format:	OpCode	
15:12	15:12	Reserved		
		Format:	MBZ	
11:0	11:0	DWord Length		
		Default Value:	20h Excludes DWord (0,1)	
		Format:	=n	
1	31:2	Reserved		
		Format:	MBZ	
1	1:0	AVC		
		Exists If:	//AVC- Decoder Only	
For AVC QM Type: This field specifies which Quantizer Matrix is loaded.				

		MFX_FQM_STATE											
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td> </tr> <tr> <td>1</td> <td>AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td> </tr> <tr> <td>2</td> <td>AVC_8x8_Intra_MATRIX</td> </tr> <tr> <td>3</td> <td>AVC_8x8_Inter_MATRIX</td> </tr> </tbody> </table>	Value	Name	0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)	1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)	2	AVC_8x8_Intra_MATRIX	3	AVC_8x8_Inter_MATRIX	
Value	Name												
0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)												
1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)												
2	AVC_8x8_Intra_MATRIX												
3	AVC_8x8_Inter_MATRIX												
	1:0	<p>MPEG2</p> <p>Exists If: //MPEG2- Decoder Only</p> <p>For MPEG2 QM Type: This field specifies which Quantizer Matrix is loaded.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>MPEG_INTRA_QUANTIZER_MATRIX</td> </tr> <tr> <td>1</td> <td>MPEG_NON_INTRA_QUANTIZER_MATRIX</td> </tr> <tr> <td>2-3</td> <td>Reserved</td> </tr> </tbody> </table>		Value	Name	0	MPEG_INTRA_QUANTIZER_MATRIX	1	MPEG_NON_INTRA_QUANTIZER_MATRIX	2-3	Reserved		
Value	Name												
0	MPEG_INTRA_QUANTIZER_MATRIX												
1	MPEG_NON_INTRA_QUANTIZER_MATRIX												
2-3	Reserved												
	1:0	<p>JPEG</p> <p>Exists If: //JPEG- Encoder Only</p> <p>For JPEG QM Type: This field specifies which Quantizer Matrix is loaded.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>JPEG_Luma_Y_QUANTIZER_MATRIX (or R)</td> </tr> <tr> <td>1</td> <td>JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)</td> </tr> <tr> <td>2</td> <td>JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j]. In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.</p>		Value	Name	0	JPEG_Luma_Y_QUANTIZER_MATRIX (or R)	1	JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)	2	JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)		
Value	Name												
0	JPEG_Luma_Y_QUANTIZER_MATRIX (or R)												
1	JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)												
2	JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)												
2..33	1023:0	<p>Forward Quantizer Matrix</p> <p>The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.</p>											



MFX_IND_OBJ_BASE_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE		
Source:	VideoCS	
Length Bias:	2	
<p>This state command provides the memory base addresses for all row stores, StreamOut buffer and reconstructed picture output buffers required by the MFD or MFC Engine (that are in addition to the row stores of the Bit Stream Decoding/Encoding Unit (BSD/BSE) and the reference picture buffers). This is a picture level state command and is common among all codec standards and for both encoder and decoder operating modes. However, some fields may only be applicable to a specific codec standard. All Pixel Surfaces (original, reference frame and reconstructed frame) in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculate the corresponding memory location within the frame buffer directly.</p> <p>The MFX_IND_OBJ_BASE_ADDR command sets the memory base address pointers for the corresponding Indirect Object Data Start Addresses (Offsets) specified in each OBJECT command. The characteristic of these indirect object data is their variable size (per MB or per Slice). Hence, each OBJECT command must specify the indirect object data offset from the base address to start fetching or writing object data.</p> <p>While the use of base address is unconditional, the indirection can be effectively disabled by setting the base address to zero.</p> <p>For decoder, there are:</p> <ul style="list-style-type: none"> • 1 read-only per-slice indirect object in the BSD_OBJECT Command, and • 2 read-only per-MB indirect objects in the IT_OBJECT Command. <p>For decoder: the Video Command Streamer (VCS) will perform the memory access bound check automatically using the corresponding MFC Indirect Object Access Upper Bound specification. If any access is at or beyond the upper bound, zero value is returned. The request to memory is still being sent, but the corresponding codec's BSD unit will detect this condition and perform the zeroing return. If the Upper Bound is turned off, the beyond bound request will return whatever is on the bus (invalid data).</p> <p>For encoder, there are:</p> <ul style="list-style-type: none"> • 1 read-only per-MB indirect object in the PAK_OBJECT Command, and • 1 write-only per-slice indirect object in the PAK_Slice_State Command <p>For encoder: whenever an out of bound address accessing request is generated, VMX will detect such requests and snap the address to the corresponding [indirect object base address + indirect data start address]. VMX will return all 0s as the data to the requestor. Notation: $PhysicalAddress[n:m]$ Corresponding bits of a physical graphics memory byte address (not mapped by a GTT); $GraphicsAddress[n:m]$ Corresponding bits of an absolute, virtual graphics memory byte address (mapped by a GTT).</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h PARALLEL_VIDEO_PIPE
		Format: OpCode

MFX_IND_OBJ_BASE_ADDR_STATE							
	28:27	Pipeline <table border="1"> <tr> <td>Default Value:</td> <td>2h MFX_IND_OBJ_BASE_ADDR_STATE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	2h MFX_IND_OBJ_BASE_ADDR_STATE	Format:	OpCode	
	Default Value:	2h MFX_IND_OBJ_BASE_ADDR_STATE					
	Format:	OpCode					
	26:24	Common Opcode <table border="1"> <tr> <td>Default Value:</td> <td>0h MFX_IND_OBJ_BASE_ADDR_STATE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h MFX_IND_OBJ_BASE_ADDR_STATE	Format:	OpCode	
	Default Value:	0h MFX_IND_OBJ_BASE_ADDR_STATE					
	Format:	OpCode					
	23:21	Sub OpcodeA <table border="1"> <tr> <td>Default Value:</td> <td>0h MFX_IND_OBJ_BASE_ADDR_STATE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h MFX_IND_OBJ_BASE_ADDR_STATE	Format:	OpCode	
	Default Value:	0h MFX_IND_OBJ_BASE_ADDR_STATE					
	Format:	OpCode					
	20:16	SubOpcodeB <table border="1"> <tr> <td>Default Value:</td> <td>3h MFX_IND_OBJ_BASE_ADDR_STATE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	3h MFX_IND_OBJ_BASE_ADDR_STATE	Format:	OpCode	
Default Value:	3h MFX_IND_OBJ_BASE_ADDR_STATE						
Format:	OpCode						
15:12	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
11:0	DWord Length <table border="1"> <tr> <td>Default Value:</td> <td>0018h Excludes DWord (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	0018h Excludes DWord (0,1)	Format:	=n		
Default Value:	0018h Excludes DWord (0,1)						
Format:	=n						
1..2	63:0 MFX Indirect Bitstream Object - Base Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_XXX_BSD_OBJECT command for fetching (reading) the compressed Slice Data. This field is only valid in MPEG2, AVC and VC1 decoder VLD mode.</p>	Format:	SplitBaseAddress4KByteAligned				
Format:	SplitBaseAddress4KByteAligned						
3	31:0 MFX Indirect Bitstream Object - Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes				
Format:	MemoryAddressAttributes						
4..5	63:0 MFX Indirect Bitstream Object - Upper Bound <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_XXX_BSD_OBJECT command for the compressed Slice Data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFX Indirect Bitstream ObjectBase Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFD_XXX_BSD_OBJECT command is set to 0. This field is only valid in MPEG2, AVC, VP8, and VC1 decoder VLD mode.</p> <table border="1"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">For VP8 Encoder, this field is corresponding to MFC Indirect PAK-BSE Object - Access Upper Bound in DW24, DW25. Please program Indirect bitstream upperbound in this field the same as DW24, DW25.</td> </tr> </table>	Format:	SplitBaseAddress4KByteAligned	Programming Notes		For VP8 Encoder , this field is corresponding to MFC Indirect PAK-BSE Object - Access Upper Bound in DW24, DW25 . Please program Indirect bitstream upperbound in this field the same as DW24, DW25.	
Format:	SplitBaseAddress4KByteAligned						
Programming Notes							
For VP8 Encoder , this field is corresponding to MFC Indirect PAK-BSE Object - Access Upper Bound in DW24, DW25 . Please program Indirect bitstream upperbound in this field the same as DW24, DW25.							
6..7	63:0 MFX Indirect MV Object - Base Address						

MFX_IND_OBJ_BASE_ADDR_STATE				
		<table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the encoder MFC_AVC_PAK_OBJECT command or the decoder MFD_IT_OBJECT command for fetching the per-MB MV data. This field is only valid in AVC encoder mode or in AVC decoder IT mode</p>	Format:	SplitBaseAddress4KByteAligned
Format:	SplitBaseAddress4KByteAligned			
8	31:0	MFX Indirect MV Object - Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
9..10	63:0	MFX Indirect MV Object - Upper Bound <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command for the per-MB MV data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFX Indirect MV Object Base Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command is set to 0. This field is only valid in AVC encoder mode or in AVC decoder IT mode.</p>	Format:	SplitBaseAddress4KByteAligned
Format:	SplitBaseAddress4KByteAligned			
11..12	63:0	MFD Indirect IT-COEFF Object - Base Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB non-scaled coefficient data (all inverse scaling and quantization are done in hardware). This field is only valid in MPEG2, AVC and VC1 decoder IT mode.</p>	Format:	SplitBaseAddress4KByteAligned
Format:	SplitBaseAddress4KByteAligned			
13	31:0	MFD Indirect IT-COEFF Object - Attributes <table border="1"> <tr> <td>Format:</td> <td>MemoryAddressAttributes</td> </tr> </table>	Format:	MemoryAddressAttributes
Format:	MemoryAddressAttributes			
14..15	63:0	MFD Indirect IT-COEFF Object - Upper Bound <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB non-scaled coefficient data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFD Indirect IT-COEFF Object Base Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect COEFF Data Length field of the MFD_IT_OBJECT command is set to 0. This field is only valid in MPEG2, AVC and VC1 decoder IT mode.</p>	Format:	SplitBaseAddress4KByteAligned
Format:	SplitBaseAddress4KByteAligned			
16..17	63:0	MFD Indirect IT-DBLK Object - Base Address <table border="1"> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB Deblocking filter control data. This field is only valid in AVC decoder IT mode.</p>	Format:	SplitBaseAddress4KByteAligned
Format:	SplitBaseAddress4KByteAligned			

MFx_IND_OBJ_BASE_ADDR_STATE		
18	31:0	MFD Indirect IT-DBLK Object - Attributes Format: MemoryAddressAttributes
19..20	63:0	MFD Indirect IT-DBLK Object - Upper Bound Format: SplitBaseAddress4KByteAligned This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB Deblocking filter control data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFD Indirect IT-DBLK Object Base Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Deblocking Control Data Length field of the MFD_IT_OBJECT command is set to 0. This field is only valid in AVC decoder IT mode.
21..22	63:0	MFC Indirect PAK-BSE Object - Base Address Format: SplitBaseAddress4KByteAligned Specifies the 4K-byte aligned memory base address for the write-only indirect data object pointed in the PAK_SLICE_STATE command for writing out the compressed bitstream. This field is only valid in AVC encoder mode.
23	31:0	MFC Indirect PAK-BSE Object - Attributes Format: MemoryAddressAttributes
24..25	63:0	MFC Indirect PAK-BSE Object - Upper Bound Format: SplitBaseAddress4KByteAligned This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the PAK_SLICE_STATE command for the per-slice output bitstream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFC Indirect PAK-BSE Object Base Address state. This field is only valid in AVC encoder mode. <div style="text-align: center; background-color: #e1eef6; padding: 5px;">Programming Notes</div> For VP8 Encoder, this field should be programmed the same at both DW4, DW5 MFC Indirect Bitstream Object - Access Upper Bound as well as DW24, DW25.

MFX_JPEG_HUFF_TABLE_STATE

MFX_JPEG_HUFF_TABLE_STATE				
Source:	VideoCS			
Length Bias:	2			
<p>This Huffman table commands contains both DC and AC tables for either luma or chroma. Once a Huffman table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used in the remaining Scans of the current image. A Huffman table will be sent to H/W only when it is loaded from bitstream.</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	Pipeline		
		Default Value:	2h MFX_MULTI_DW	
	26:24	Media Command Opcode		
		Default Value:	7h JPEG_COMMON	
	23:21	SubOpcode A		
		Default Value:	0h	
	20:16	SubOpcode B		
Default Value:		2h		
15:12	Reserved			
11:0	DWord Length			
	Default Value:	033Dh Excludes DWord (0,1)		
	Format:	=n		
1	31:1	Reserved		
	Format:	MBZ		
	0	HuffTableID (1-bit) Identifies the huffman table.		
		Value	Name	Description
		0	Y	Huffman table for Y
2..4	95:0	DC_BITS (12 8-bit array) The number of DC Huffman codes of length i, where i is 1~12		

MFx_JPEG_HUFF_TABLE_STATE				
5..7	95:0	DC_HUFFVAL (12 8-bit array) The value associated with each DC Huffman code of length i.		
8..11	127:0	AC_BITS (16 8-bit array) the list of L_i , number of Huffman codes of length i, where i is 1~16		
12..51	1279:0	AC_HUFFVAL (160 8-bit array) the list of $V_{i,j}$, the value associated with each Huffman code of length i		
52	31:16	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px; height: 15px;"></td><td style="width: 50px; text-align: center;">MBZ</td></tr></table>		MBZ
		MBZ		
15:0	AC_HUFFVAL(2-8 bit array) In AC table, BITS can have up to 16-bit codeword. L_i can be 0 ~ 162. HUFFVAL will have a list of likely random distributed values			

MFX_JPEG_PIC_STATE

MFX_JPEG_PIC_STATE			
Source:		VideoCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_MULTI_DW
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	7h JPEG
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		0h Common	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	0h MEDIA_	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length	Format:	=n
1	31	Reserved	
		Format:	MBZ
	30:26	Pixels In Horizontal Last MCU	
Exists If:		//Encoder Only	
		The number of pixels in the last MCU in a row MCUs. This information is used for completion of partial MCU.	
25:21	Pixels In Vertical Last MCU		
	Exists If:	//Encoder Only	
		The number of pixels in the last MCU in a column MCUs. This information is used for completion	

MFX_JPEG_PIC_STATE

		of partial MCU.													
20	Vertical Up-Sampling Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td colspan="2">//Decoder Only</td> </tr> </table> <p>Only applied to chroma blocks. This flag is used for 2:1 vertical up-sampling for chroma 420 and outputting chroma422 YUY2 or UYVY format. To enable this flag, the input should be interleaved Scan, InputFormatYUV should be set to YUV420, and OutputFormatYUV should be set to YUY2 or UYVY.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td></td> <td>no up-sampling</td> </tr> <tr> <td>1b</td> <td></td> <td>2:1 vertical up-sampling</td> </tr> </tbody> </table>			Exists If:	//Decoder Only		Value	Name	Description	0b		no up-sampling	1b		2:1 vertical up-sampling
Exists If:	//Decoder Only														
Value	Name	Description													
0b		no up-sampling													
1b		2:1 vertical up-sampling													
19	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td colspan="2">MBZ</td> </tr> </table>			Format:	MBZ										
Format:	MBZ														
18	Horizontal Down-Sampling Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td colspan="2">//Decoder Only</td> </tr> </table> <p>Only applied to chroma blocks. This flag is used for 2:1 horizontal down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, InputFormatYUV should be set to YUV422V_2Y or YUV422V_4Y, and OutputFormatYUV should be set to NV12.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td></td> <td>no down-sampling</td> </tr> <tr> <td>1b</td> <td></td> <td>2:1 horizontal down-sampling</td> </tr> </tbody> </table>			Exists If:	//Decoder Only		Value	Name	Description	0b		no down-sampling	1b		2:1 horizontal down-sampling
Exists If:	//Decoder Only														
Value	Name	Description													
0b		no down-sampling													
1b		2:1 horizontal down-sampling													
17	Vertical Down-Sampling Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td colspan="2">//Decoder Only</td> </tr> </table> <p>Only applied to chroma blocks. This flag is used for 2:1 vertical down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, InputFormatYUV should be set to YUV422H_2Y or YUV422H_4Y, and OutputFormatYUV should be set to NV12.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td></td> <td>no down-sampling</td> </tr> <tr> <td>1b</td> <td></td> <td>2:1 vertical down-sampling</td> </tr> </tbody> </table>			Exists If:	//Decoder Only		Value	Name	Description	0b		no down-sampling	1b		2:1 vertical down-sampling
Exists If:	//Decoder Only														
Value	Name	Description													
0b		no down-sampling													
1b		2:1 vertical down-sampling													
16	Average Down Sampling <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td colspan="2">//Decoder Only</td> </tr> </table> <p>This flag is used to select a down-sampling method when VertDownSamplingEnb or HoriDownSamplingEnb is set to 1.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td></td> <td>Drop every other line (or column) pixels</td> </tr> <tr> <td>1b</td> <td></td> <td>Average neighboring two pixels</td> </tr> </tbody> </table>			Exists If:	//Decoder Only		Value	Name	Description	0b		Drop every other line (or column) pixels	1b		Average neighboring two pixels
Exists If:	//Decoder Only														
Value	Name	Description													
0b		Drop every other line (or column) pixels													
1b		Average neighboring two pixels													
15:12	Reserved														

MFX_JPEG_PIC_STATE

	Format:	MBZ	
11:8	Input Surface Format YUV		
	Exists If:	//Encoder Only	
	This field specifies the surface format to read a YUV image data		
	Value	Name	Description
	0000b		Reserved
	0001b	NV12	NV12 for chroma 4:2:0
	0010b	UYVY	UYVY for chroma 4:2:2
	0011b	YUY2	YUY2 for chroma 4:2:2
	0100b	Y8	Y8 for chroma400 Y-only image
	0101b	RGB	RGB or YUV for chroma 4:4:4
Programming Notes			
This field should be set accordingly for SurfaceFormat in MFX_SURFACE_STATE command.			
R8G8B8A8_UNORM in this field is used for encoding RGB and YUV chroma 4:4:4. For RGB input, any order of image components R, G, B (e.g., RGB, GBR, BGR, YUV) will be acceptable as far as the order of Quantization tables and Huffman tables match the order of image components.			
11:8	Output Format YUV		
	Exists If:	//Decoder Only	
	This field specifies the surface format to write the decoded JPEG image. Note that any non-interleaved JPEG input should be set to "0000". For the interleaved input Scan data, it can be set either "0000" or the corresponding format.		
	Value	Name	Description
	0000b		3 separate plane for Y, U, and V respectively
	0001b		NV12 for chroma 4:2:0
	0010b		UYVY for chroma 4:2:2
	0011b		YUY2 for chroma 4:2:2
	Programming Notes		
	The MFX_SURFACE_STATE command should be set accordingly for each OutputFormatYUV . For NV12, Surface Format = 4 (PLANAR_420_8) For YUY2, Surface Format = 0 (YCRCB_NORMAL) For UYVY, Surface Format = 3 (YCRCB_SWAPY) NV12 (0001b) can be set only when Y, U, V are interleaved in a single Scan data with the following cases		
<ul style="list-style-type: none"> • InputFormatYUV is YUV420 and VertDownSamplingEnb is disabled • InputFormatYUV is YUV422H_2Y or YUV422H_4Y, and VertDownSamplingEnb is enabled 			
UYVY (0010b) and YUY2 (0011b) can be set only when Y, U, V are interleaved in a single Scan			

MFX_JPEG_PIC_STATE

		data with the following cases <ul style="list-style-type: none"> • InputFormatYUV is YUV420 and VertUpSamplingEnb is enabled • InputFormatYUV is YUV422H_2Y or YUV422H_4Y and VertUpSamplingEnb is disabled 																																
7:6	Reserved	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>		Format:	MBZ																													
Format:	MBZ																																	
5:4	Rotation	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Decoder Only</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td></td> <td>no rotation</td> </tr> <tr> <td>01b</td> <td></td> <td>rotate clockwise 90 degree</td> </tr> <tr> <td>10b</td> <td></td> <td>rotate counter-clockwise 90 degree (same as rotating 270 degree clockwise)</td> </tr> <tr> <td>11b</td> <td></td> <td>rotate 180 degree (NOT the same as flipped on the x-axis)</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr style="background-color: #e6f2ff;"> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Rotation can be set to 01b, 10b, or 11b when OutputFormatYUV is set to 0000b. For other OutputFormatYUV, Rotation is not allowed.</td> </tr> </tbody> </table>		Exists If:	//Decoder Only	Value	Name	Description	00b		no rotation	01b		rotate clockwise 90 degree	10b		rotate counter-clockwise 90 degree (same as rotating 270 degree clockwise)	11b		rotate 180 degree (NOT the same as flipped on the x-axis)	Programming Notes		Rotation can be set to 01b, 10b, or 11b when OutputFormatYUV is set to 0000b. For other OutputFormatYUV, Rotation is not allowed.											
Exists If:	//Decoder Only																																	
Value	Name	Description																																
00b		no rotation																																
01b		rotate clockwise 90 degree																																
10b		rotate counter-clockwise 90 degree (same as rotating 270 degree clockwise)																																
11b		rotate 180 degree (NOT the same as flipped on the x-axis)																																
Programming Notes																																		
Rotation can be set to 01b, 10b, or 11b when OutputFormatYUV is set to 0000b. For other OutputFormatYUV, Rotation is not allowed.																																		
3	Reserved	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>		Format:	MBZ																													
Format:	MBZ																																	
2:0	Input Format YUV	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>U3</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td style="text-align: center;">[Default]</td> <td>YUV400 (grayscale image)</td> </tr> <tr> <td>1</td> <td></td> <td>YUV420</td> </tr> <tr> <td>2</td> <td></td> <td>YUV422H_2Y (Horizontally chroma 2:1 subsampled) - horizontal 2 Y-block, 1U and 1V</td> </tr> <tr> <td>3</td> <td></td> <td>YUV444</td> </tr> <tr> <td>4</td> <td></td> <td>YUV411</td> </tr> <tr> <td>5</td> <td></td> <td>YUV422V_2Y (Vertically chroma 2:1 subsampled) - vertical 2 Y-blocks, 1U and 1V</td> </tr> <tr> <td>6</td> <td></td> <td>YUV422H_4Y - 2x2 Y-blocks, vertical 2U and 2V</td> </tr> <tr> <td>7</td> <td></td> <td>YUV422V_4Y - 2x2 Y-blocks, horizontal 2U and 2V</td> </tr> </tbody> </table>		Exists If:	//Decoder Only	Format:	U3	Value	Name	Description	0	[Default]	YUV400 (grayscale image)	1		YUV420	2		YUV422H_2Y (Horizontally chroma 2:1 subsampled) - horizontal 2 Y-block, 1U and 1V	3		YUV444	4		YUV411	5		YUV422V_2Y (Vertically chroma 2:1 subsampled) - vertical 2 Y-blocks, 1U and 1V	6		YUV422H_4Y - 2x2 Y-blocks, vertical 2U and 2V	7		YUV422V_4Y - 2x2 Y-blocks, horizontal 2U and 2V
Exists If:	//Decoder Only																																	
Format:	U3																																	
Value	Name	Description																																
0	[Default]	YUV400 (grayscale image)																																
1		YUV420																																
2		YUV422H_2Y (Horizontally chroma 2:1 subsampled) - horizontal 2 Y-block, 1U and 1V																																
3		YUV444																																
4		YUV411																																
5		YUV422V_2Y (Vertically chroma 2:1 subsampled) - vertical 2 Y-blocks, 1U and 1V																																
6		YUV422H_4Y - 2x2 Y-blocks, vertical 2U and 2V																																
7		YUV422V_4Y - 2x2 Y-blocks, horizontal 2U and 2V																																
2:0	Output MCU Structure	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="width: 60%;">//Encoder Only</td> </tr> </table>		Exists If:	//Encoder Only																													
Exists If:	//Encoder Only																																	

MFX_JPEG_PIC_STATE																												
	<p>Output MCU Structure(OutputMcuStructure) should be set accordingly for each Input Surface Format YUV(InputSurfaceFormatYUV):</p> <ul style="list-style-type: none"> • If InputSurfaceFormatYUV is set to NV12, OutputMCUStructure is set to YUV420. • If InputSurfaceFormatYUV is set to UYVY or YUY2, OutputMCUStructure is set to YUV422H_2Y. • If InputSurfaceFormatYUV is set to Y8, OutputMCUStructure is set to YUV400. • If InputSurfaceFormatYUV is set to RGB (or GBR, BGR, YUV), OutputMCuStructure is set to RGB. • If InputSurfaceFormatYUV is set to RGB, the order of encoded blocks in MCU will be same as the order of input image components. If the order of input image components is RGB (or GBR, BGR, YUV), then the order of blocks will be RGB (or GBR, BGR, YUV respectively). <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>YUV400</td> <td>Grayscale Image</td> </tr> <tr> <td style="text-align: center;">1</td> <td>YUV420</td> <td>Both horizontally and vertically chroma 2:1 subsampled</td> </tr> <tr> <td style="text-align: center;">2</td> <td>YUV422H_2Y</td> <td>Horizontally chroma 2:1 subsampled - horizontal 2 Y-blocks, 1 U and 1 V block</td> </tr> <tr> <td style="text-align: center;">3</td> <td>RGB</td> <td>RGB or YUV444: No subsample</td> </tr> <tr> <td style="text-align: center;">4</td> <td></td> <td></td> </tr> <tr> <td style="text-align: center;">5</td> <td></td> <td></td> </tr> <tr> <td style="text-align: center;">6</td> <td></td> <td></td> </tr> <tr> <td style="text-align: center;">7</td> <td></td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0	YUV400	Grayscale Image	1	YUV420	Both horizontally and vertically chroma 2:1 subsampled	2	YUV422H_2Y	Horizontally chroma 2:1 subsampled - horizontal 2 Y-blocks, 1 U and 1 V block	3	RGB	RGB or YUV444: No subsample	4			5			6			7		
Value	Name	Description																										
0	YUV400	Grayscale Image																										
1	YUV420	Both horizontally and vertically chroma 2:1 subsampled																										
2	YUV422H_2Y	Horizontally chroma 2:1 subsampled - horizontal 2 Y-blocks, 1 U and 1 V block																										
3	RGB	RGB or YUV444: No subsample																										
4																												
5																												
6																												
7																												
2	<p>31:30 Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table> <p>29 Output Pixel Normalize</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="text-align: center;">//Decoder Only</td> </tr> </table> <p>JPEG decoded output pixels for Y and U/V in order to adjust display YUV range.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> <th style="text-align: center;">Exists If</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td></td> <td>No Normalization</td> <td></td> </tr> <tr> <td style="text-align: center;">1</td> <td></td> <td>Normalize output pixels from [0,255] to [16,235]</td> <td style="text-align: center;">//Y</td> </tr> <tr> <td style="text-align: center;">1</td> <td></td> <td>Normalize output pixels from [0,255] to [16,239]</td> <td style="text-align: center;">//U/V</td> </tr> </tbody> </table> <p>28:16 Frame Height In Blocks Minus 1</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Exists If:</td> <td style="text-align: center;">//Decoder Only</td> </tr> <tr> <td>Format:</td> <td style="text-align: center;">U13-1</td> </tr> </table> <p>(The number of blocks in height) - 1. This value is calculated using the number of lines Y and vertical sampling factor of the first component V₁ in Frame header. See the note following this table. For interleaved components, $((Y + (V_1 * 8 - 1)) / (V_1 * 8)) * V_1 - 1$, where "/" is integer division. For non-interleaved components, $((Y + 7) / 8) - 1$.</p>	Format:	MBZ	Exists If:	//Decoder Only	Value	Name	Description	Exists If	0		No Normalization		1		Normalize output pixels from [0,255] to [16,235]	//Y	1		Normalize output pixels from [0,255] to [16,239]	//U/V	Exists If:	//Decoder Only	Format:	U13-1			
Format:	MBZ																											
Exists If:	//Decoder Only																											
Value	Name	Description	Exists If																									
0		No Normalization																										
1		Normalize output pixels from [0,255] to [16,235]	//Y																									
1		Normalize output pixels from [0,255] to [16,239]	//U/V																									
Exists If:	//Decoder Only																											
Format:	U13-1																											

MFX_JPEG_PIC_STATE

28:16	Frame Height In Blks Minus 1	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U13-1</td> </tr> </table>	Exists If:	//Encoder Only	Format:	U13-1																									
Exists If:	//Encoder Only																														
Format:	U13-1																														
<p>(The number of blocks in height) - 1. This value is calculated using the number of lines Y and vertical sampling factor of the first component V1 in Frame header. See the note following this table.</p> <p>For interleaved components: $((Y + (V1 * 8 - 1)) / (V1 * 8)) * V1 - 1$ For non-interleaved components: $((Y + 7) / 8) - 1$</p>																															
15:13	RoundingQuant	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Encoder Only</td> </tr> </table> <p>Rounding value applied to quantization output</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 25%;">Value</th> <th style="width: 25%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>000b</td> <td>[Default]</td> <td>1/2</td> </tr> <tr> <td>001b</td> <td></td> <td>$(1/2 - 1/128)$</td> </tr> <tr> <td>010b</td> <td></td> <td>$(1/2 + 1/128)$</td> </tr> <tr> <td>011b</td> <td></td> <td>$(1/2 - 1/64)$</td> </tr> <tr> <td>100b</td> <td></td> <td>$(1/2 + 1/64)$</td> </tr> <tr> <td>101b</td> <td></td> <td>$(1/2 - 1/32)$</td> </tr> <tr> <td>110b</td> <td></td> <td>$(1/2 - 1/16)$</td> </tr> <tr> <td>111b</td> <td></td> <td>$(1/2 - 1/8)$</td> </tr> </tbody> </table>	Exists If:	//Encoder Only	Value	Name	Description	000b	[Default]	1/2	001b		$(1/2 - 1/128)$	010b		$(1/2 + 1/128)$	011b		$(1/2 - 1/64)$	100b		$(1/2 + 1/64)$	101b		$(1/2 - 1/32)$	110b		$(1/2 - 1/16)$	111b		$(1/2 - 1/8)$
Exists If:	//Encoder Only																														
Value	Name	Description																													
000b	[Default]	1/2																													
001b		$(1/2 - 1/128)$																													
010b		$(1/2 + 1/128)$																													
011b		$(1/2 - 1/64)$																													
100b		$(1/2 + 1/64)$																													
101b		$(1/2 - 1/32)$																													
110b		$(1/2 - 1/16)$																													
111b		$(1/2 - 1/8)$																													
12:0	Frame Width In Blocks Minus 1	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>U13-1</td> </tr> </table> <p>(The number of blocks in width) - 1. This value is calculated using the number of samples per line X and horizontal sampling factor of the first component H₁ in Frame header. See the note following this table. For interleaved components, $((X + (H_1 * 8 - 1)) / (H_1 * 8)) * H_1 - 1$. For non-interleaved components, $((X + 7) / 8) - 1$.</p>	Exists If:	//Decoder Only	Format:	U13-1																									
Exists If:	//Decoder Only																														
Format:	U13-1																														
12:0	Frame Width In Blks Minus 1	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U13-1</td> </tr> </table> <p>(The number of blocks in width) - 1. This value is calculated using the number of samples per line X and horizontal sampling factor of the first component H₁ in Frame header. See the note following this table.</p> <p>For interleaved components: $((X + (H_1 * 8 - 1)) / (H_1 * 8)) * H_1 - 1$ For non-interleaved components: $((X + 7) / 8) - 1$</p>	Exists If:	//Encoder Only	Format:	U13-1																									
Exists If:	//Encoder Only																														
Format:	U13-1																														

MFX_MPEG_TS_CONTROL command

MFX_MPEG_TS_CONTROL command			
Source:		VideoCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h Command Type
		Format:	Opcode
	28:27	Pipeline	
		Default Value:	2h
		Format:	Opcode
	26:24	Opcode	
		Default Value:	0h
		Format:	Opcode
	23:21	SubOpA	
Default Value:		2h	
Format:		Opcode	
20:16	SubOpB		
	Default Value:	Bh	
	Format:	Opcode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Value	Name	Description
	3h	DWORD_COUNT_n [Default]	Total length - 2 (excludes DWord0 and DWord1)
1	31:30	Reserved	
		Format:	MBZ
	29	payload_unit_start_indicator control	
		Programming Notes	
			This bit should be programmed zero always.
	28	Additional Copy info flag in PES header	
	27	DSM trick mode flag in PES header	
	26	Original or flag in PES header	
	25	Copy Right flag in PES header	
	24	Output TS packet grouping select	
0: Return all packets continuously			

MFX_MPEG_TS_CONTROL command				
		1: Return 7 packets in 2K aligned buffer (with the remaining bits between the end of the 7 th packet and the end of the 2K buffer including the rest being undefined)		
	23:20	StreamID lower Nibble Stream ID Lower Nibble. Stream ID for Video can be 0xE0 through 0xEF. This 4 bit field indicates the last 4 bits of Stream ID in Mpeg transport stream as indicated in the DCN diagram		
	19:13	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">MBZ</td></tr></table>		MBZ
	MBZ			
	12:0	Video PacketID Header Parameter This field specifies the static program fields in MPEG header for each Video packet.		
2	31:0	PCR 90 KHz component least significant bits.		
3	31:23	27MHz Counter Full 8-bits of 27Mhz counter		
	22:1	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">MBZ</td></tr></table>		MBZ
		MBZ		
0	90 KHz counter MSB Upper bit (bit 33) of 90khz counter			
4	31:0	PTS Delta PTS Delta to be applied to 90 KHz count of PCR to generate PTS. This is a Twos complement number and added to 90 KHz PCR counter to generate PTS.		
5	31:28	Continuity Counter This field specifies the 4b continuity counter given in the MPEGTS packet header. This should be initialized with the value that was read from MMIO at the end of the previous frame. That value will be incremented by HW so there is no need to SW to increment it. For the first frame this should be set to 0.		
	27:16	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"> </td><td style="width: 50px;">MBZ</td></tr></table>		MBZ
		MBZ		
15:0	MPEGTS Packet Count This field is ignored by HW. Driver can copyMFX_PAK_MPEGTS_STATUS register from the previous frame to DW 5 ofMPEG_TS_CONTROL_command usingMI_STORE_REG_MEM			

MFX_MPEG2_PIC_STATE

MFX_MPEG2_PIC_STATE				
Source:	VideoCS			
Length Bias:	2			
<p>This must be the very first command to issue after the surface state, the pipe select and base address setting commands. For MPEG-2 the encoder is called per slice-group, however the picture state is called per picture. Notice that a slice-group is a group of consecutive slices that no non-trivial slice headers are inserted in between.</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	28:27	Pipeline	
			Default Value:	2h MFX_MPEG2_PIC_STATE
			Format:	OpCode
	26:24	26:24	Media Command Opcode	
			Default Value:	3h MPEG2_COMMON
			Format:	OpCode
	23:21	23:21	SubOpcode A	
Default Value:			0h	
Format:			OpCode	
20:16	20:16	SubOpcode B		
		Default Value:	0h	
		Format:	OpCode	
15:12	15:12	Reserved		
		Format:	MBZ	
11:0	11:0	DWord Length		
		Default Value:	0h Excludes DWord (0,1)= 00Bh, used for normal decode and encode mode000h, a special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware.	
		Format:	=n	
1	31:28	f_code[1][1]. Used for backward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details		
	27:24	f_code[1][0]. Used for backward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details		
	23:20	f_code[0][1] Used for forward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details		

MFX_MPEG2_PIC_STATE													
19:16	f_code[0][0]	Used for forward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details											
15:14	Intra DC Precision	<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> See ISO/IEC 13818-2 6.3.10 for details.	Format:	U2									
Format:	U2												
13:12	Picture Structure	This field specifies whether the picture is encoded in the form of a frame picture or one field (top or bottom) picture. See ISO/IEC 13818-2 6.3.10 for details. Format = MPEG_PICTURE_STRUCTURE00 = Reserved01 = MPEG_TOP_FIELD10 = MPEG_BOTTOM_FIELD11 = MPEG_FRAME											
11	TFF (Top Field First)	When two fields are stored in a picture, this bit indicates if the top field is the first field. For a frame P picture, the value 1 indicates that the top field of the reconstructed frame is the first field output by the decoding process, the same as defined in ISO/IEC 13818-2 6.3.10. Particularly, it is used by the hardware to calculate derivative motion vectors from the dual-prime motion vectors. For a field P picture, hardware uses this bit together with the Picture Structure to determine if the current picture is the Second Field. In this case, the definition of this bit differs from ISO/IEC 13818-2 6.3.10 - software must derive the value for this bit according to the following relation: Picture Structure = top field Picture Structure = bottom field Second Field = 0 TFF = 1 TFF = 0 Second Field = 1 TFF = 0 TFF = 1											
10	Frame Prediction Frame DCT	This field provides constraints on the DCT type and prediction type. It affects the syntax of the bitstream.											
9	Concealment Motion Vector Flag	This field indicates if the concealment motion vectors are coded in intra macroblocks. It affects the syntax of the bitstream.											
8	Quantizer Scale Type	<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Boolean</td> </tr> </table> This field specifies the quantizer scaling type. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td></td> <td>MPEG_QSCALE_LINEAR</td> </tr> <tr> <td>1h</td> <td></td> <td>D MPEG_QSCALE_NONLINEAR esc</td> </tr> </tbody> </table>	Format:	Boolean	Value	Name	Description	0h		MPEG_QSCALE_LINEAR	1h		D MPEG_QSCALE_NONLINEAR esc
Format:	Boolean												
Value	Name	Description											
0h		MPEG_QSCALE_LINEAR											
1h		D MPEG_QSCALE_NONLINEAR esc											
7	Intra VLC Format	This field is used by VLD											
6	Scan Order	<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Boolean</td> </tr> </table> This field specifies the Inverse Scan method for the DCT-domain coefficients in the blocks of the current picture. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td></td> <td>MPEG_ZIGZAG_SCAN</td> </tr> </tbody> </table>	Format:	Boolean	Value	Name	Description	0h		MPEG_ZIGZAG_SCAN			
Format:	Boolean												
Value	Name	Description											
0h		MPEG_ZIGZAG_SCAN											

MFX_MPEG2_PIC_STATE																	
		1h MPEG_ALTERNATE_VERTICAL_SCAN															
	5:0	Reserved Format: MBZ															
2	31	I Slice Concealment Mode Exists If: //Decoder This field controls how MPEG decoder handles MB concealment in I Slice															
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Intra Concealment</td> <td>Using Coefficient values to handle MB concealment</td> </tr> <tr> <td>1h</td> <td>Inter Concealment</td> <td>Using Motion Vectors to handle MB concealment</td> </tr> </tbody> </table>	Value	Name	Description	0h	Intra Concealment	Using Coefficient values to handle MB concealment	1h	Inter Concealment	Using Motion Vectors to handle MB concealment						
		Value	Name	Description													
		0h	Intra Concealment	Using Coefficient values to handle MB concealment													
		1h	Inter Concealment	Using Motion Vectors to handle MB concealment													
Programming Notes																	
If this field is set to "1", driver must provide a valid forward reference picture (both top and bottom Field must be valid)																	
	30	Reserved Format: MBZ															
	29:28	P/B Slice Concealment Mode Exists If: //Decoder This field controls how MPEG decoder handles MB concealment in P/B Slice.															
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>INTER</td> <td>If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use forward reference (same polarity for field pic) with MV final values set to 0.</td> </tr> <tr> <td>01b</td> <td>LEFT</td> <td>If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)</td> </tr> <tr> <td>10b</td> <td>ZERO</td> <td>Always use forward reference (same polarity for field pic) with MV final values set to 0 (Macroblock is concealed as INTER coded)</td> </tr> <tr> <td>11b</td> <td>INTRA</td> <td>Use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)</td> </tr> </tbody> </table>	Value	Name	Description	00b	INTER	If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use forward reference (same polarity for field pic) with MV final values set to 0.	01b	LEFT	If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)	10b	ZERO	Always use forward reference (same polarity for field pic) with MV final values set to 0 (Macroblock is concealed as INTER coded)	11b	INTRA	Use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)
Value	Name	Description															
00b	INTER	If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use forward reference (same polarity for field pic) with MV final values set to 0.															
01b	LEFT	If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)															
10b	ZERO	Always use forward reference (same polarity for field pic) with MV final values set to 0 (Macroblock is concealed as INTER coded)															
11b	INTRA	Use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data)															
	27	Reserved Format: MBZ															
	26:25	P/B Slice Predicted BiDir Motion Type Override - Bi-direction MV Type Override Exists If: //Decoder This field is only applicable if the Concealment Motion Type is predicted to be Bi-directional. (It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB is a bi-directional MB).															

MFX_MPEG2_PIC_STATE

Value	Name	Description
0h	BID	Keep Bi-direction Prediction
1h	RESERVED	
2h	FWD	Only use Forward Prediction (Backward MV is forced to invalid
3h	BWD	Only use Backward Prediction (Forward MV is forced to invalid)
24	P/B Slice Predicted Motion Vector Override Final MV value Override	
Exists If:		//Decoder
This field is only applicable if the Concealment Motion Vectors are non-zero. It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB has non-zero motion vectors).		
Value	Name	Description
0h	Predicted	Motion Vectors use predicted values
1h	ZERO	Motion Vectors force to 0
23:15	Reserved	
Format:		MBZ
14	LoadSlicePointerFlag - LoadBitStreamPointerPerSlice	
Exists If:		//Encoder
To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.		
Value	Name	Description
0h		Load BitStream Pointer only once for the first slice of a frame
1h		Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field
13:11	Reserved	
Format:		MBZ
10:9	Picture Coding Type	
Format:		U2
This field identifies whether the picture is an intra-coded picture (I), predictive-coded picture (P) or bi-directionally predictive-coded picture (B). See ISO/IEC 13818-2 6.3.9 for details.		
Value	Name	
00b	Reserved	
01b	MPEG_I_PICTURE	
10b	10 = MPEG_P_PICTURE	

MFX_MPEG2_PIC_STATE																	
	11b	MPEG_B_PICTURE															
	8:2	Reserved Format: _____ MBZ															
	1:0	MismatchControlDisabled These 2 bits flag disables mismatch control of the inverse transformation for some specific cases during reference reconstruction. To disable MPEG2 IDCT fixed point arithmetic correction. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td></td> <td>Mismatch control applies to all MBs</td> </tr> <tr> <td>01b</td> <td></td> <td>Disable mismatch control to all intra MBs whose all AC-coefficients are zero.</td> </tr> <tr> <td>10b</td> <td></td> <td>Disable mismatch control to all MBs whose all AC-coefficients are zero.</td> </tr> <tr> <td>11b</td> <td></td> <td>Disable mismatch control to all MBs.</td> </tr> </tbody> </table>	Value	Name	Description	00b		Mismatch control applies to all MBs	01b		Disable mismatch control to all intra MBs whose all AC-coefficients are zero.	10b		Disable mismatch control to all MBs whose all AC-coefficients are zero.	11b		Disable mismatch control to all MBs.
Value	Name	Description															
00b		Mismatch control applies to all MBs															
01b		Disable mismatch control to all intra MBs whose all AC-coefficients are zero.															
10b		Disable mismatch control to all MBs whose all AC-coefficients are zero.															
11b		Disable mismatch control to all MBs.															
3	31	Slice Concealment Disable Bit Exists If: _____ //Decode If VINunit detects the next slice starting position is either out-of-bound or smaller than or equal to the current slice starting position, VIN will set the current slice to be 1 MB and force VMDunit to do slice concealment on the next slice. This bit will disable this feature and the MB data from the next slice will be decoded from bitstream. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Enable [Default]</td> <td>VIN will force next slice to be concealment if detects slice boundary error</td> </tr> <tr> <td>1h</td> <td>Disable</td> <td>VIN will not force next slice to be in concealment</td> </tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <p style="text-align: center;">Programming Notes</p> <p>Driver has an option to detect the scenario given in description (above) and remove the second (out-of-order) slice. In this case, hardware will decode the first slice in completion and do concealment till the third slice. It should yield a picture with better quality this way.</p> </div>	Value	Name	Description	0h	Enable [Default]	VIN will force next slice to be concealment if detects slice boundary error	1h	Disable	VIN will not force next slice to be in concealment						
Value	Name	Description															
0h	Enable [Default]	VIN will force next slice to be concealment if detects slice boundary error															
1h	Disable	VIN will not force next slice to be in concealment															
	30:29	Reserved Format: _____ MBZ															
	28:24	Reserved															
	23:16	FrameHeightInMBsMinus1[7:0] (Picture Height in Macroblocks) Format: _____ U8															
	15:8	Reserved Format: _____ MBZ															
	7:0	FrameWidthInMBsMinus1[7:0] (Picture Width in Macroblocks) Format: _____ U8															
4	31:16	MinFrameWSize															

MFX_MPEG2_PIC_STATE

		Format:	U16
		<p>Minimum Frame Size [15:0] (16-bit) (Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control. Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax (DWORD 10 bits 29:16). This field is reserved in Decode mode.</p>	
		Programming Notes	
		Programmable range is 0..(2 ¹⁶ -1).	
	15	Reserved	
		Format:	MBZ
	14:12	RoundInterAC , rounding precision for non-Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16	
	11	Reserved	
		Format:	MBZ
	10:8	RoundIntraAC	
		Format:	U3
		rounding precision for Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16	
	7	Reserved	
		Format:	MBZ
	6:4	RoundInterDC rounding Precision for non-Intra-DC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16	
	3	Reserved	
		Format:	MBZ
	2:1	RoundIntraDC rounding Precision for Intra-DC00: +1/801: +2/810: +3/811: +4/8	
	0	Reserved	
		Format:	MBZ
5	31:17	Reserved	
		Format:	MBZ
	16	FrameSizeControlMask Frame size conformance mask. This field is used when MacroblockStatEnable is set to 1.	
		Value	Name Description
		0h	Do not change Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for

MFX_MPEG2_PIC_STATE		
		frame level Rate control
	1h	Replace Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control values in MFC_IMAGE_STATUS control register.
15:13	Reserved	
	Format:	MBZ
12	InterMBForceCBPZeroControlMask	
	Format:	U1
	Inter MB Force CBP ZERO mask.	
	Value	Name Description
	0h	No effect
	1h	Zero out all A/C coefficients for the inter MB violating Inter Conformance
11:10	MinFrameWSizeUnits	
	This field is the Minimum Frame Size Units	
	Value	Name Description
	00b	compatibility mode Minimum Frame Size is in old mode (words, 2bytes)
	01b	16 byte Minimum Frame Size is in 16bytes
	10b	4Kb Minimum Frame Size is in 4Kbytes
	11b	16Kb Minimum Frame Size is in 16Kbytes
9	MBRateControlMask	
	MB Rate Control conformance mask. This field is ignored when MacroblockStatEnable is disabled or MB level Rate control flag for the current MB is disable in Macroblock Status Buffer.	
	Value	Name Description
	0h	Do not change QP values of inter macroblock with suggested QP values in Macroblock Status Buffer
	1h	Apply RC QP delta for all macroblock
8:4	Reserved	
	Format:	MBZ
3	FrameBitRateMinReportMask	
	This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin.	
	Value	Name Description
	0h	Disable Do not update bit0 of MFC_IMAGE_STATUS control register.
	1h	Enable set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit.
2	FrameBitRateMaxReportMask	
	This is a mask bit controlling if the condition of frame level bit count exceeds	

MFX_MPEG2_PIC_STATE											
		<p>FrameBitRateMax.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.	1h	Enable	set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.
Value	Name	Description									
0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.									
1h	Enable	set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.									
	1	<p>InterMBMaxSizeReportMask This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td></td> <td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td> </tr> <tr> <td>1h</td> <td></td> <td>set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.</td> </tr> </tbody> </table>	Value	Name	Description	0h		Do not update bit0 of MFC_IMAGE_STATUS control register.	1h		set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.
Value	Name	Description									
0h		Do not update bit0 of MFC_IMAGE_STATUS control register.									
1h		set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.									
	0	<p>IntraMBMaxSizeReportMask This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td></td> <td>Do not update bit0 of MFC_IMAGE_STATUS control register.</td> </tr> <tr> <td>1h</td> <td></td> <td>set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.</td> </tr> </tbody> </table>	Value	Name	Description	0h		Do not update bit0 of MFC_IMAGE_STATUS control register.	1h		set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.
Value	Name	Description									
0h		Do not update bit0 of MFC_IMAGE_STATUS control register.									
1h		set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.									
6 [ExistsIf]Encode Only	31:28	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
	27:16	<p>InterMBMaxSize</p> <table border="1"> <tr> <td>Default Value:</td> <td>FFFh</td> </tr> </table> <p>This field, Inter MB Conformance Max size limit, indicates the allowed max bit count size for Inter MB</p>	Default Value:	FFFh							
	Default Value:	FFFh									
15:12	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
11:0	<p>IntraMBMaxSize</p> <table border="1"> <tr> <td>Default Value:</td> <td>FFFh</td> </tr> </table> <p>This field, Intra MB Conformance Max size limit, indicates the allowed max bit count size for Intra MB</p>	Default Value:	FFFh								
Default Value:	FFFh										
7	31:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ										
8 [ExistsIf]Encode	31:24	<p>SliceDeltaQPMax[3]</p> <table border="1"> <tr> <td>Format:</td> <td>S7</td> </tr> </table>	Format:	S7							
Format:	S7										

MFX_MPEG2_PIC_STATE

Only		<p>This field is the Slice level delta QP for total bit-count above FrameBitRateMax - first 1/8 region. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of (FrameBitRateMax, (FrameBitRateMax+FrameBitRateMaxDelta»3).</p> <p>Range: [-30,30]</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">Disable</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">Enable</td> </tr> </tbody> </table>	Value	Name	0h	Disable	1h	Enable
Value	Name							
0h	Disable							
1h	Enable							
	23:16	<p>SliceDeltaQPMax[2]</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%; text-align: center;">S7</td> </tr> </table> <p>Range: [-30,30]</p> <p>This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/8 and below 1/4 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta»3), (FrameBitRateMax+FrameBitRateMaxDelta»2).</p>	Format:	S7				
Format:	S7							
	15:8	<p>SliceDeltaQPMax[1]</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%; text-align: center;">S7</td> </tr> </table> <p>Range: [-30,30]</p> <p>This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/4 and below 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between and of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+FrameBitRateMaxDelta»2), (FrameBitRateMax+FrameBitRateMaxDelta»1).</p>	Format:	S7				
Format:	S7							
	7:0	<p>SliceDeltaQPMax[0]</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%; text-align: center;">S7</td> </tr> </table> <p>Range: [-30,30]</p> <p>This field is the Slice level delta QP for bit-count above FrameBitRateMax - above 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta , i.e., in the range of ((FrameBitRateMax+FrameBitRateMaxDelta»1), infinite).</p>	Format:	S7				
Format:	S7							
9	31:24	<p>SliceDeltaQPMin[3]</p>						

MFX_MPEG2_PIC_STATE								
[ExistsIf]Encode Only		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table>	Format:	S7				
	Format:	S7						
		<p>Range: [-30,30]</p> <p>This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 region. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3), \text{FrameBitRateMin}]$.</p>						
	23:16	<p>SliceDeltaQPMin[2]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table>	Format:	S7				
Format:	S7							
	<p>Range: [-30,30]</p> <p>This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/8 and above 1/4. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3)]$.</p>							
	<p>15:8</p>	<p>SliceDeltaQPMin[1]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table>	Format:	S7				
Format:	S7							
	<p>Range: [-30,30]</p> <p>This field is the Slice level delta QP for bit-count below FrameBitRateMin - below 1/4 and above 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 1), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2)]$.</p>							
	<p>7:0</p>	<p>SliceDeltaQPMin[0]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table>	Format:	S7				
Format:	S7							
	<p>Range: [-30,30]</p> <p>This field is the Slice Level Delta QP for bit-count below FrameBitRateMin - below 1/2. This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bitcount for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta, i.e., in the range of $[0, (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 1)]$.</p>							
10 [ExistsIf]Encode Only	31	<p>FrameBitrateMaxUnit</p> <p>This field is the Frame Bitrate Maximum Limit Units.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e6f2ff;"> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table>	Value	Name	Description			
Value	Name	Description						

MFX_MPEG2_PIC_STATE

	0h	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0
	1h	Kilobyte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0
30	FrameBitrateMaxUnitMode BitField This field is the Frame Bitrate Maximum Limit Units.dDesc		
	Value	Name	Description
	0h	Compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)
	1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)
29:16	FrameBitRateMax This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28 and 29 should be 0.		
	Value	Name	Description
	[0-16383]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
	[0-16383]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
15	FrameBitrateMinUnit This field is the Frame Bitrate Minimum Limit Units.		
	Value	Name	Description
	0h	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0
	1h	KiloByte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0
14	FrameBitrateMinUnitMode This field is the Frame Bitrate Minimum Limit Units.ValueNameDescription		
	Value	Name	Description
	0h	compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)
	1h	New Mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)
13:0	FrameBitRateMin This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines minimum allowed bits in a frame before multi-pass gets triggered (when		

MFX_MPEG2_PIC_STATE

		<p>enabled). In other words, multi-pass is triggered when the actual frame byte count happen to be below this value. It takes on a value in the range of [0-16383]. When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0. WhenFrameBitrateMinUnit=0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit. WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.</p>		
11 [ExistsIf]Encode Only	31	Reserved		
		Format:	MBZ	
	30:16	FrameBitRateMaxDelta		
		Default Value:	0h	
		Access:	None	
		Format:	U15	
		<p>This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode), only bits 16:27 should be used, bits 28, 29 and 30 should be 0.</p> <p>This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0. Range : [0-32767] WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit. WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.</p>		
	15	Reserved		
		Format:	MBZ	
	14:0	FrameBitRateMinDelta		
		<p>This field is used to select the slice delta QPwhen FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. WhenFrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits12, 13 and 14 should be 0.Note: HW requires the following conditionFrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may causeunpredicted behavior.</p>		
		Value	Name	Description
		[0-32767]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.

MFX_MPEG2_PIC_STATE			
		[0-32767]	WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.
12	31:0	Reserved	
		Format:	MBZ

MFX_PAK_INSERT_OBJECT

MFX_PAK_INSERT_OBJECT		
Source:	VideoCS	
Length Bias:	2	
<p>The MFX_PAK_INSERT_OBJECT command is the first primitive command for the AVC, MPEG2, JPEG, and VP8 Encoding Pipeline.</p> <p>This command is issued to setup the control and parameters of inserting a chunk of compressed/encoded bits into the current bitstream output buffer starting at the specified bit location to perform the actual insertion by transferring the command inline data to the output buffer max, 32 bits at a time.</p> <p>It is a variable length command as the data to be inserted are presented as inline data of this command. It is a multiple of 32-bit (1 DW), as the data bus to the bitstream buffer is 32-bit wide.</p> <p>Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid H.264 bitstream.</p> <p>Internally, MFX hardware will keep track of the very last two bytes' (the very last byte can be a partial byte) values of the previous insertion. It is required that the next Insertion Object Command or the next PAK Object Command to perform the start code emulation sequence check and prevention 0x03 byte insertion with this end condition of the previous insertion.</p> <p>Hardware will keep track of an output bitstream buffer current byte position and the associated next bit insertion position index. Data to be inserted can be a valid H.264 NAL units or a partial NAL unit. Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by STATE Command) determines the number of CABAC_ZERO_WORD to be inserted to the end of the current NAL, based on the minimum byte size of a NAL and the actual bin count of the encoded Slice. Since prior to the CABAC_ZERO_WORD insertion, the RBSP or EBSP is already byte-aligned, so each CABAC_ZERO_WORD insertion is actually a 3-byte sequence 0x00 00 03. The inline data may have already been processed for start code emulation byte insertion, except the possibility of the last 2 bytes plus the very last partial byte (if any). Hence, when hardware performing the concatenation of multiple consecutive insertion commands, or concatenation of an insertion command and a PAK object command, it must check and perform the necessary start code emulation byte insert at the junction. The inline data is required to be byte aligned on the left (first transmitted bit order) and may or may not be byte aligned on the right (last transmitted bits).</p> <p>The command will specify the bit offset of the last valid DW. Each insertion state command defines a chunk of bits (compressed data) to be inserted at a specific location of the output compressed bitstream in the output buffer. Depend on CABAC or CAVLC encoding mode (from Slice State), PAK Object Command is always ended in byte aligned output bitstream except for CABAC header insertion which is bit aligned. In the aligned cases, PAK will perform 0 filling in CAVLC mode, and 1 filling in CABAC mode.</p> <p>Insertion data can include: any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current Slice SPS NALPPS NALSEI NAL Other Non-Slice NAL Leading_Zero_8_bits (as many bytes as there is) Start Code Prefix NAL Header Byte Slice Header Any encoded syntax elements bit data after the encoded Slice Data (PAK Object Command) of the current Slice and prior to the next encoded Slice Data of the next Slice or prior to the end of the bistream, whichever comes first Cabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is).</p> <p>Anything listed above before a Slice Data Context switch interrupt is not supported by this command.</p>		
DWord	Bit	Description

MFX_PAK_INSERT_OBJECT													
0	31:29	Command Type <table border="1"> <tr> <td>Default Value:</td> <td>3h PARALLEL_VIDEO_PIPE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	3h PARALLEL_VIDEO_PIPE	Format:	OpCode							
	Default Value:	3h PARALLEL_VIDEO_PIPE											
	Format:	OpCode											
	28:27	Pipeline <table border="1"> <tr> <td>Default Value:</td> <td>2h MFX_PAK_INSERT_OBJECT</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	2h MFX_PAK_INSERT_OBJECT	Format:	OpCode							
	Default Value:	2h MFX_PAK_INSERT_OBJECT											
	Format:	OpCode											
	26:24	Media Command Opcode <table border="1"> <tr> <td>Default Value:</td> <td>0h MFX_COMMON</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h MFX_COMMON	Format:	OpCode							
	Default Value:	0h MFX_COMMON											
Format:	OpCode												
23:21	SubOpcode A <table border="1"> <tr> <td>Default Value:</td> <td>2h</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	2h	Format:	OpCode								
Default Value:	2h												
Format:	OpCode												
20:16	SubOpcode B <table border="1"> <tr> <td>Default Value:</td> <td>8h</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	8h	Format:	OpCode								
Default Value:	8h												
Format:	OpCode												
15:12	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
11:0	DWord Length <table border="1"> <tr> <td>Default Value:</td> <td>[1h, FFFh] Excludes DWord (0,1) = Variable Length in DW</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	[1h, FFFh] Excludes DWord (0,1) = Variable Length in DW	Format:	=n								
Default Value:	[1h, FFFh] Excludes DWord (0,1) = Variable Length in DW												
Format:	=n												
1	31:18	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
	Format:	MBZ											
	17:16	DataByteOffset - SrcDataStartingByteOffset[1:0] Source Data Starting Byte Position within the very first inline DW. <table border="1"> <tr> <th colspan="3" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="3">Must be set to 0 for JPEG encoder</td> </tr> </table>	Programming Notes			Must be set to 0 for JPEG encoder							
	Programming Notes												
Must be set to 0 for JPEG encoder													
15	HeaderLengthExcludeFrmSize In case this flag is on, bits are NOT accumulated during current access unit coding neither for Cabac Zero Word insertion bits counting or for output in MMIO register MFC_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER. When using HeaderLenghtExcludeFrmSize for header insertion, the software needs to make sure that data comes already with inserted start code emulation bytes. SW shouldn't set EmulationFlag bit (Bit 3 of DWORD1 of MFX_PAK_INSERT_OBJECT). <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>NO_ACCUMULATION</td> <td>Bits during current call are not accumulated</td> </tr> <tr> <td>0</td> <td>ACCUMULATE</td> <td>All bits accumulated</td> </tr> </tbody> </table> <table border="1"> <tr> <th colspan="3" style="text-align: center;">Programming Notes</th> </tr> </table>	Value	Name	Description	1	NO_ACCUMULATION	Bits during current call are not accumulated	0	ACCUMULATE	All bits accumulated	Programming Notes		
Value	Name	Description											
1	NO_ACCUMULATION	Bits during current call are not accumulated											
0	ACCUMULATE	All bits accumulated											
Programming Notes													
<table border="1"> <tr> <th colspan="3" style="text-align: center;">Programming Notes</th> </tr> </table>			Programming Notes										
Programming Notes													

MFX_PAK_INSERT_OBJECT

		Must be set to 0 for JPEG encoder										
14	<p>Slice Header Indicator</p> <p>This bit indicates if the insert object is a slice header. In the VDEnc mode, PAK only gets this command at the beginning of the frame for slice position X=0, Y=0. It internally generates the header that needs to be inserted per slice. For VDEnc mode, this bit should always be set.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>SLICE_HEADER</td> <td>Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.</td> </tr> <tr> <td style="text-align: center;">0</td> <td>LEGACY</td> <td>Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>In VDENC mode, we support only Slice layer without partitioning RBSP syntax. The payload for PAK_INS_OBJ should contain only start code for Slice header followed by NAL_type and slice header (slice_header() in AVC spec).The payload for PAK_INS_OBJ shouldn't contain CABAC Byte alignment bits. HW adds these alignment bits which are part of slice_data. Example PAK_INS_OBJ payload : 00 00 01 <NAL_type> <slice_header_Byte0> ..<slice_header_Byte LAST>Any zero_bytes that are added before slice header can be inserted by any preceding general PAK_INS_OBJ.</p>			Value	Name	Description	1	SLICE_HEADER	Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.	0	LEGACY	Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.
Value	Name	Description										
1	SLICE_HEADER	Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.										
0	LEGACY	Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.										
13:8	<p>DataBitsInLastDW - SrCDataEndingBitInclusion[5:0]</p> <p>Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first. For example, SrCDataEndingBitInclusion = 9, bit 7:0 and bit 15 are included as valid header data.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[1,32]</td> <td></td> </tr> </tbody> </table>			Value	Name	[1,32]						
Value	Name											
[1,32]												
7:4	<p>SkipEmulByteCnt - Skip Emulation Byte Count</p> <p>Skip emulation check for number of starting bytes. It can be programmed from 0 to 15 bytes. For example, to skip the start code that has already prefixed in the bitstream.</p> <p style="text-align: center;">Programming Notes</p> <p>Must be set to 0 for JPEG encoder</p>											
3	<p>EmulationFlag - EmulationByteBitsInsertEnable</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>NONE</td> <td>No emulation</td> </tr> <tr> <td style="text-align: center;">1</td> <td>EMULATE</td> <td>Instruct the hardware to perform Start Code Prefix (0x 00 00 01/02/03/00) Search and Prevention Byte (0x 03) insertion on the insertion data of this command. It is required that hardware will handle a start code prefix crossing the boundary between insertion commands, or an insertion command followed by a PAK Object command.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Must be set to 0 for JPEG encoder</p>			Value	Name	Description	0	NONE	No emulation	1	EMULATE	Instruct the hardware to perform Start Code Prefix (0x 00 00 01/02/03/00) Search and Prevention Byte (0x 03) insertion on the insertion data of this command. It is required that hardware will handle a start code prefix crossing the boundary between insertion commands, or an insertion command followed by a PAK Object command.
Value	Name	Description										
0	NONE	No emulation										
1	EMULATE	Instruct the hardware to perform Start Code Prefix (0x 00 00 01/02/03/00) Search and Prevention Byte (0x 03) insertion on the insertion data of this command. It is required that hardware will handle a start code prefix crossing the boundary between insertion commands, or an insertion command followed by a PAK Object command.										

MFX_PAK_INSERT_OBJECT											
	2	<p>Las theaderFlag - LastSrcHeaderDataInsertCommandFlag</p> <p>To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series. In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command. In CAVLC, hardware ignores this bit</p>									
	1	<p>EndOfSliceFlag - LastDstDataInsertCommandFlag</p> <p>No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory</p>									
	0	<p>BitstreamStartReset - ResetBitStreamStartingPos</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">RESET</td> <td>Reset the bitstream buffer insertion position to the bitstream buffer starting position.</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">INSERT</td> <td>Insert the current command inline data starting at the current bitstream buffer insertion position</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Must be set to 1 for JPEG encoder</p>	Value	Name	Description	1	RESET	Reset the bitstream buffer insertion position to the bitstream buffer starting position.	0	INSERT	Insert the current command inline data starting at the current bitstream buffer insertion position
	Value	Name	Description								
	1	RESET	Reset the bitstream buffer insertion position to the bitstream buffer starting position.								
0	INSERT	Insert the current command inline data starting at the current bitstream buffer insertion position									
2..n	31:0	<p>Insert Data PayLoad</p> <p>Actual Data to be inserted to the output bitstream buffer.</p>									

MFX_PIPE_BUF_ADDR_STATE

MFX_PIPE_BUF_ADDR_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This state command provides the memory base addresses for all row stores, StreamOut buffer and reconstructed picture output buffers required by the MFD or MFC Engine (that are in addition to the row stores of the Bit Stream Decoding/Encoding Unit (BSD/BSE) and the reference picture buffers).</p> <p>This is a picture level state command and is common among all codec standards and for both encoder and decoder operating modes. However, some fields may only applicable to a specific codec standard. All Pixel Surfaces (original, reference frame and reconstructed frame) in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculated the corresponding memory location within the frame buffer directly.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_PIPE_BUF_ADDR_STATE
		Format:	OpCode
	26:24	Common Opcode	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	2h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length	Format:	=n
		Fixed Length	
	Value	Name	Description
	3Fh	DWORD_COUNT_n [Default]	Excludes DWord (0,1)
1	31:6	Pre Deblocking Destination Address	

MFX_PIPE_BUF_ADDR_STATE																	
		<table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:6]</td> </tr> </table> <p>Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit). This field is ignored if PreDeblockOutEnable is set to 0 (disable).</p>	Format:	GraphicsAddress[31:6]													
Format:	GraphicsAddress[31:6]																
	5:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
2	31:16	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
15:0	<p>Pre Deblocking Destination Address High</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> <p>This field is for the upper range of Pre-Deblocking Destination Address. This field is ignored if PreDeblockOutEnable is set to 0 (disable).</p>	Format:	GraphicsAddress[47:32]														
Format:	GraphicsAddress[47:32]																
3	31:15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
	14:13	<p>Pre Deblocking - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
1h	TRMODE_TILEYF	4KB tiled resources															
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
12:11	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
10	<p>Compression Type This field is valid only is Memory Compression is enabled.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled										
Value	Name																
0	Media Compression Enabled [Default]																
1	Render Compression Enabled																
9	<p>Pre Deblocking - Memory Compression Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Memory compression will be attempted for this surface.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p>	Format:	Enable	Value	Name	0	Compression Disable	1	Compression Enable								
Format:	Enable																
Value	Name																
0	Compression Disable																
1	Compression Enable																

MFX_PIPE_BUF_ADDR_STATE

		Video Mode	Compression Enable	
		AVC Frame Only (No MBAFF or Field)	Yes	
		VP8 (Only Frame is supported)	Yes	
		VC1 (No overlap smoothing, No field)	Yes	
		MPGE2 (No field)	Yes	
JPEG Decode				
		Chroma Format	Output Format	Compression Enable
		422H_2Y,422H_4Y	YUY2	Yes
		422H_2Y,422H_4Y	YUY2	Yes
		422H_2Y,422H_4Y	UYVY	Yes
		422H_2Y, 422H_4Y, 422V_2Y, 422V_4Y	NV12	No
		420	YUY2, UYVY	No
		420	NV12	Yes
	8:7	Pre Deblocking - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.		
		Value	Name	
		00b	Highest priority	
		01b	Second highest priority	
		10b	Third highest priority	
		11b	Lowest priority	
	6:0	Pre Deblocking - Memory Object Control State		
		Format:	MEMORY_OBJECT_CONTROL_STATE	
		Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).		
4	31:6	Post Deblocking Destination Address		
		Format:	GraphicsAddress[31:6]	
		Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)This field is ignored if PostDeblockOutEnable is set to 0 (disable).		
	5:0	Reserved		
		Format:	MBZ	
5	31:16	Reserved		
		Format:	MBZ	
	15:0	Post Deblocking Destination Address High		
		Format:	GraphicsAddress[47:32]	
		This field is for the upper range of Post-Deblocking Destination Address. This field is ignored if		

MFX_PIPE_BUF_ADDR_STATE																	
		PostDeblockOutEnable is set to 0 (disable).															
6	31:15	Reserved Format: _____ MBZ															
	14:13	Post Deblocking - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 45%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
	1h	TRMODE_TILEYF	4KB tiled resources														
	2h	TRMODE_TILEYS	64KB tiled resources														
3h	Reserved																
12:11	Reserved Format: _____ MBZ																
10	Compression Type This field is applicable only when Memory compression is enabled. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled										
Value	Name																
0	Media Compression Enabled [Default]																
1	Render Compression Enabled																
9	Post Deblocking - Memory Compression Enable Format: _____ Enable Memory compression will be attempted for this surface. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable	1	Compression Enable										
Value	Name																
0	Compression Disable																
1	Compression Enable																
8:7	Post Deblocking - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 85%;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority						
Value	Name																
00b	Highest priority																
01b	Second highest priority																
10b	Third highest priority																
11b	Lowest priority																
6:0	Post Deblocking - Memory Object Control State Format: _____ MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).																
7	31:6 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: left;">Original Uncompressed Picture Source Address</th> </tr> </thead> <tbody> <tr> <td style="width: 30%;">Format:</td> <td>GraphicsAddress[31:6]</td> </tr> </tbody> </table>	Original Uncompressed Picture Source Address		Format:	GraphicsAddress[31:6]												
Original Uncompressed Picture Source Address																	
Format:	GraphicsAddress[31:6]																

MFX_PIPE_BUF_ADDR_STATE																	
		Specifies the 64 byte aligned frame buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding. This field is only valid in encoding mode.															
	5:0	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
8	31:16	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
15:0	Original Uncompressed Picture Source Address High <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> <p>This field is for the upper range of Original Uncompressed Picture Source Address. This field is valid for encoding mode only.</p>	Format:	GraphicsAddress[47:32]														
Format:	GraphicsAddress[47:32]																
9	31:15	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
	14:13	Original Uncompressed Picture - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode. <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 40%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
0h	TRMODE_NONE	No tiled resource															
1h	TRMODE_TILEYF	4KB tiled resources															
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
12:11	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
10	Compression Type <table border="1" style="width: 100%; text-align: center;"> <tr> <th colspan="2">Description</th> </tr> <tr> <td colspan="2">This field is valid only when memory compression enable is true.</td> </tr> </table> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Description		This field is valid only when memory compression enable is true.		Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled						
Description																	
This field is valid only when memory compression enable is true.																	
Value	Name																
0	Media Compression Enabled [Default]																
1	Render Compression Enabled																
9	Original Uncompressed Picture - Memory Compression Enable Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 70%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable												
Value	Name																
0	Compression Disable																
8:7	Original Uncompressed Picture Source - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 70%;">Name</th> </tr> </thead> <tbody> </tbody> </table>	Value	Name														
Value	Name																

MFX_PIPE_BUF_ADDR_STATE																	
		<table border="1"> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </table>	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority							
00b	Highest priority																
01b	Second highest priority																
10b	Third highest priority																
11b	Lowest priority																
	6:0	<p>Original Uncompressed Picture Source - Memory Object Control State</p> <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).</p>	Format:	MEMORY_OBJECT_CONTROL_STATE													
Format:	MEMORY_OBJECT_CONTROL_STATE																
10	31:6	<p>StreamOut Data Destination Base Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:6]</td> </tr> </table> <p>Specifies the 64 byte aligned address for outputting the per-MB indirect data to memory when StreamOutEnable is set in the MFX_PIPE_MODE_SELECT command. For Decoder : This field is used for transcoding purpose. For Encoder : This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.</p>	Format:	GraphicsAddress[31:6]													
Format:	GraphicsAddress[31:6]																
	5:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
11	31:16	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	15:0	<p>StreamOut Data Destination Base Address High</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> <p>This field is for the upper range of Original Uncompressed Picture Source Address</p>	Format:	GraphicsAddress[47:32]													
Format:	GraphicsAddress[47:32]																
12	31:15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	14:13	<p>StreamOut Data Destination - Tiled Resource Mode For Media Surfaces:This field specifies the tiled resource mode.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
Value	Name	Description															
0h	TRMODE_NONE	No tiled resource															
1h	TRMODE_TILEYF	4KB tiled resources															
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
	12:10	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	9	<p>StreamOut Data Destination - Memory Compression Enable</p>															

MFX_PIPE_BUF_ADDR_STATE												
		<p>Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Compression Disable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable						
Value	Name											
0	Compression Disable											
	8:7	<p>StreamOut Data Destination - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td>Highest priority</td> </tr> <tr> <td style="text-align: center;">01b</td> <td>Second highest priority</td> </tr> <tr> <td style="text-align: center;">10b</td> <td>Third highest priority</td> </tr> <tr> <td style="text-align: center;">11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority
Value	Name											
00b	Highest priority											
01b	Second highest priority											
10b	Third highest priority											
11b	Lowest priority											
	6:0	<p>StreamOut Data Destination - Memory Object Control State</p> <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).</p>	Format:	MEMORY_OBJECT_CONTROL_STATE								
Format:	MEMORY_OBJECT_CONTROL_STATE											
13	31:6	<p>Intra Row Store Scratch Buffer Base Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:6]</td> </tr> </table> <p>This field provides the base address of the scratch buffer (read/write) used by the AVC/VP8 IntraPrediction unit to store MB information of the previous row for processing of each macroblock in the current row. The Intra Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Intra Row Store. This field is ignored in MPEG2 and VC1 mode. Max 256 cachelines for 4K pixels (1 cacheline for either MBAFF or non-MBAFF) Intra Row Store Scratch Buffer - Arbitration Priority Control</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td> <p>This is one of the four RowStore Scratch Buffers which can be programmed to use the internal Media Cache (total size 640 CacheLine). When Intra Row Store Scratch Buffer Cache Select is programmed to "1", this data will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address to Media Cache. Driver needs to make sure the whole buffer fits into MFX Media Internal Storage.</p> <p><i>(Notes: 1 cacheline per MB, and the buffer needs to have enough space for 1 MB row).</i></p> </td> </tr> </tbody> </table>	Format:	GraphicsAddress[31:6]	Programming Notes	<p>This is one of the four RowStore Scratch Buffers which can be programmed to use the internal Media Cache (total size 640 CacheLine). When Intra Row Store Scratch Buffer Cache Select is programmed to "1", this data will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address to Media Cache. Driver needs to make sure the whole buffer fits into MFX Media Internal Storage.</p> <p><i>(Notes: 1 cacheline per MB, and the buffer needs to have enough space for 1 MB row).</i></p>						
Format:	GraphicsAddress[31:6]											
Programming Notes												
<p>This is one of the four RowStore Scratch Buffers which can be programmed to use the internal Media Cache (total size 640 CacheLine). When Intra Row Store Scratch Buffer Cache Select is programmed to "1", this data will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address to Media Cache. Driver needs to make sure the whole buffer fits into MFX Media Internal Storage.</p> <p><i>(Notes: 1 cacheline per MB, and the buffer needs to have enough space for 1 MB row).</i></p>												
	5:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
14	31:16	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	15:0	<p>Intra Row Store Scratch Buffer Base Address High</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> <p>This field is for the upper range of Intra RowStore/Scratch Buffer Base Address. This field is</p>	Format:	GraphicsAddress[47:32]								
Format:	GraphicsAddress[47:32]											

MFX_PIPE_BUF_ADDR_STATE																
		ignored in MPEG2 and VC1 mode.														
15	31:15	Reserved Format: MBZ														
	14:13	Intra Row Store Scratch Buffer - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved
Value	Name	Description														
0h	TRMODE_NONE	No tiled resource														
1h	TRMODE_TILEYF	4KB tiled resources														
2h	TRMODE_TILEYS	64KB tiled resources														
3h	Reserved															
12	Intra Row Store Scratch Buffer Cache Select This field controls if Intra Row Store is going to store inside Media Cache or to LLC.															
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>Buffer going to LLC.</td> </tr> <tr> <td>1</td> <td></td> <td>Buffer going to Internal Media Storage</td> </tr> </tbody> </table>		Value	Name	Description	0		Buffer going to LLC.	1		Buffer going to Internal Media Storage					
	Value	Name	Description													
0		Buffer going to LLC.														
1		Buffer going to Internal Media Storage														
Reserved Format: MBZ																
10	Reserved - Intra Row Store															
9	Intra Row Store Scratch Buffer - Memory Compression Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> </tbody> </table>		Value	Name	0	Compression Disable										
	Value	Name														
	0	Compression Disable														
Programming Notes																
This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed																
8:7	Intra Row Store Scratch Buffer - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.															
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </tbody> </table>		Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority				
	Value	Name														
	00b	Highest priority														
	01b	Second highest priority														
10b	Third highest priority															
11b	Lowest priority															
6:0	Intra Row Store Scratch Buffer - Memory Object Control State Format: MEMORY_OBJECT_CONTROL_STATE															
	Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).															

MFX_PIPE_BUF_ADDR_STATE																	
16	31:6	Deblocking Filter Row Store Scratch Base Address Format: GraphicsAddress[31:6] Deblocking Filter Row Store is needed for: <ul style="list-style-type: none"> • AVC and VC1 In-Loop Deblocking Filter • VC1 Overlap-smoothing Filter • AVC, VC1, and MPEG-2 Out-Of-Loop Deblocking Filter (Intel extension) This field provides the 64 byte aligned base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store. Max 6 cachelines for VC1 and MPEG2, and max 4 for AVC (for MBAFF, 2 for non-MBAFF)															
		<div style="text-align: center;">Programming Notes</div> This is one of the four RowStore Scratch Buffers which can be programmed to use the internal Media Cache (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage. <i>(Notes: 2 cachelines per MB for non-mbaff; 4 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).</i>															
17	5:0	Reserved Format: MBZ															
	31:16	Reserved Format: MBZ															
18	15:0	Deblocking Filter Row Store Scratch Base Address High Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Filter Row Store Scratch Buffer Address.															
	31:15	Reserved Format: MBZ															
18	14:13	Deblocking Filter Row Store - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode.															
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
		Value	Name	Description													
		0h	TRMODE_NONE	No tiled resource													
		1h	TRMODE_TILEYF	4KB tiled resources													
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
12	Deblocking Filter Row Store Scratch Buffer Cache Select																

MFX_PIPE_BUF_ADDR_STATE													
		<p>This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>Buffer going to LLC</td> </tr> <tr> <td>1</td> <td></td> <td>Buffer going to Media Internal Storage</td> </tr> </tbody> </table>	Value	Name	Description	0		Buffer going to LLC	1		Buffer going to Media Internal Storage		
Value	Name	Description											
0		Buffer going to LLC											
1		Buffer going to Media Internal Storage											
	11	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	10	<p>Deblocking Filter Row Store Scratch - Memory Compression Mode</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved [Default]</td> </tr> </tbody> </table>	Value	Name	0	Reserved [Default]							
Value	Name												
0	Reserved [Default]												
	9	<p>Deblocking Filter Row Store Scratch - Memory Compression Enable</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed</p>	Value	Name	0	Compression Disable							
Value	Name												
0	Compression Disable												
	8:7	<p>Deblocking Filter Row Store Scratch - Arbitration Priority Control</p> <p>This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority	
Value	Name												
00b	Highest priority												
01b	Second highest priority												
10b	Third highest priority												
11b	Lowest priority												
	6:0	<p>Deblocking Filter Row Store Scratch - Memory Object Control State</p> <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).</p>	Format:	MEMORY_OBJECT_CONTROL_STATE									
Format:	MEMORY_OBJECT_CONTROL_STATE												
19..50	1023:0	<p>Reference Picture Base Addr</p> <table border="1"> <tr> <td>Format:</td> <td>MFX_REFERENCE_PICTURE_BASE_ADDR[16]</td> </tr> </table>	Format:	MFX_REFERENCE_PICTURE_BASE_ADDR[16]									
Format:	MFX_REFERENCE_PICTURE_BASE_ADDR[16]												
51	31:15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
	Format:	MBZ											
14:13	<p>Reference Picture - Tiled Resource Mode</p> <p>For Media Surfaces:This field specifies the tiled resource mode.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources
Value	Name	Description											
0h	TRMODE_NONE	No tiled resource											
1h	TRMODE_TILEYF	4KB tiled resources											
2h	TRMODE_TILEYS	64KB tiled resources											

MFX_PIPE_BUF_ADDR_STATE												
	3h	Reserved										
12:9	Reserved Format: _____ MBZ											
8:7	Reference Picture - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest priority</td> </tr> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </tbody> </table>		Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority
Value	Name											
00b	Highest priority											
01b	Second highest priority											
10b	Third highest priority											
11b	Lowest priority											
6:0	Reference Picture - Memory Object Control State Format: _____ MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).											
52	31:6	Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address Format: _____ GraphicsAddress[31:6] For decoder: Specifies the 64 byte aligned buffer address for writing a single error/status record into the memory when Pic Error/Status Report Enable is set in the MFX_PIPE_MODE_SELECT Command. The error/status record is written by HW at the end of decoding one single picture. The record is written in a fixed format, total 96-bits in size always. Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format. For encoder: Specifies the 64 byte aligned buffer address for reading the per-MB indirect data from memory when MacroblockStatEnable is set in the MFX_AVC_IMG_STATE Command. This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit, and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.										
	5:0	Reserved Format: _____ MBZ										
53	31:16	Reserved Format: _____ MBZ										
	15:0	Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address High Format: _____ GraphicsAddress[47:32] This field is for the upper range of Macroblock Status Buffer Base Address										
54	31:15	Reserved										

MFX_PIPE_BUF_ADDR_STATE

		Format:	MBZ
14:13	Macroblock Status Buffer - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode.		
	Value	Name	Description
	0h	TRMODE_NONE	No tiled resource
	1h	TRMODE_TILEYF	4KB tiled resources
	2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved		
12:11	Reserved		
	Format:	MBZ	
10	Macroblock Status Buffer - Memory Compression Mode		
	Value	Name	
	0	Reserved [Default]	
9	Macroblock Status Buffer - Memory Compression Enable		
	Value	Name	
	0	Compression Disable	
	Programming Notes		
This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed			
8:7	Macroblock Status Buffer - Arbitration Priority Control		
	This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.		
	Value	Name	
	00b	Highest priority	
	01b	Second highest priority	
10b	Third highest priority		
11b	Lowest priority		
6:0	Macroblock Status Buffer - Memory Object Control State		
	Format:	MEMORY_OBJECT_CONTROL_STATE	
Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).			
55	31:6	Macroblock ILDB StreamOut Buffer Base Address	
		Format:	GraphicsAddress[31:6]
Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when Debocker streamout enable is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.			

MFX_PIPE_BUF_ADDR_STATE																	
	5:0	Reserved															
		Format: MBZ															
56	31:16	Reserved															
		Format: MBZ															
	15:0	Macroblock ILDB StreamOut Buffer Base Address High Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Filter Row Store Scratch Address															
57	31:15	Reserved															
		Format: MBZ															
	14:13	Macroblock ILDB StreamOut - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode.															
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td style="text-align: center;">2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td style="text-align: center;">3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
		Value	Name	Description													
		0h	TRMODE_NONE	No tiled resource													
		1h	TRMODE_TILEYF	4KB tiled resources													
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
12:11	Reserved																
	Format: MBZ																
10	Macroblock ILDB StreamOut Buffer - Memory Compression Mode																
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Reserved [Default]</td> </tr> </tbody> </table>	Value	Name	0	Reserved [Default]												
Value	Name																
0	Reserved [Default]																
9	Macroblock ILDB StreamOut Buffer - Memory Compression Enable																
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Compression Disable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable												
	Value	Name															
0	Compression Disable																
<p style="text-align: center;">Programming Notes</p> This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed																	
8:7	Macroblock ILDB StreamOut Buffer - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.																
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td>Highest priority</td> </tr> <tr> <td style="text-align: center;">01b</td> <td>Second highest priority</td> </tr> <tr> <td style="text-align: center;">10b</td> <td>Third highest priority</td> </tr> <tr> <td style="text-align: center;">11b</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest priority	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority						
	Value	Name															
	00b	Highest priority															
	01b	Second highest priority															
10b	Third highest priority																
11b	Lowest priority																
6:0	Macroblock ILDB StreamOut Buffer - Memory Object Control State																

MFX_PIPE_BUF_ADDR_STATE																	
		<table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).</p>	Format:	MEMORY_OBJECT_CONTROL_STATE													
Format:	MEMORY_OBJECT_CONTROL_STATE																
58	31:6	<p>Second Macroblock ILDB StreamOut Buffer Base Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:6]</td> </tr> </table> <p>64 byte aligned buffer. Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when Debocker streamout enable is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.</p>	Format:	GraphicsAddress[31:6]													
	Format:	GraphicsAddress[31:6]															
5:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
59	31:16	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
15:0	<p>Second Macroblock ILDB StreamOut Buffer Base Address High</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> <p>This field is for the upper range of Second Macroblock ILDB StreamOutBuffer Base Address.</p>	Format:	GraphicsAddress[47:32]														
Format:	GraphicsAddress[47:32]																
60	31:15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
	14:13	<p>Second Macroblock ILDB StreamOut Buffer - Tiled Resource Mode For Media Surfaces: This field specifies the tiled resource mode.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
1h	TRMODE_TILEYF	4KB tiled resources															
2h	TRMODE_TILEYS	64KB tiled resources															
3h	Reserved																
12:11	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
10	<p>Second Macroblock ILDB StreamOut Buffer - Memory Compression Mode</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved [Default]</td> </tr> </tbody> </table>	Value	Name	0	Reserved [Default]												
Value	Name																
0	Reserved [Default]																
9	<p>Second Macroblock ILDB StreamOut Buffer - Memory Compression Enable</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p>	Value	Name	0	Compression Disable												
Value	Name																
0	Compression Disable																

MFX_PIPE_BUF_ADDR_STATE

		This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed						
	8:7	Second Macroblock ILDB StreamOut Buffer - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.						
	6:0	Second Macroblock ILDB StreamOut Buffer - Memory Object Control State Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).						
61	31	Reference Picture 15 - Compression Type This field is valid only when memory compression is enabled.						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled
		Value	Name					
	0	Media Compression Enabled [Default]						
	1	Render Compression Enabled						
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable	1	Compression Enable	
	Value	Name						
	0	Compression Disable						
	1	Compression Enable						
	30	Reference Picture 15 - Memory Compression Enable						
<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table>		Value	Name	0	Compression Disable	1	Compression Enable	
Value		Name						
0	Compression Disable							
1	Compression Enable							
<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled		
Value	Name							
0	Media Compression Enabled [Default]							
1	Render Compression Enabled							
29	Reference Picture 14 - Compression Type							
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled	
	Value	Name						
0	Media Compression Enabled [Default]							
1	Render Compression Enabled							
<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable	1	Compression Enable		
Value	Name							
0	Compression Disable							
1	Compression Enable							
28	Reference Picture 14 - Memory Compression Enable							
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable	1	Compression Enable	
	Value	Name						
0	Compression Disable							
1	Compression Enable							
<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled		
Value	Name							
0	Media Compression Enabled [Default]							
1	Render Compression Enabled							
27	Reference Picture 13 - Compression Type							
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled	
	Value	Name						
0	Media Compression Enabled [Default]							
1	Render Compression Enabled							
<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable	1	Compression Enable		
Value	Name							
0	Compression Disable							
1	Compression Enable							
26	Reference Picture 13 - Memory Compression Enable							
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Compression Disable</td> </tr> <tr> <td>1</td> <td>Compression Enable</td> </tr> </tbody> </table>	Value	Name	0	Compression Disable	1	Compression Enable	
	Value	Name						
0	Compression Disable							
1	Compression Enable							
<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Media Compression Enabled [Default]</td> </tr> <tr> <td>1</td> <td>Render Compression Enabled</td> </tr> </tbody> </table>	Value	Name	0	Media Compression Enabled [Default]	1	Render Compression Enabled		
Value	Name							
0	Media Compression Enabled [Default]							
1	Render Compression Enabled							
25	Reference Picture 12 - Compression Type							

MFX_PIPE_BUF_ADDR_STATE

		Value	Name
		0	Media Compression Enabled [Default]
		1	Render Compression Enabled
24	Reference Picture 12 - Memory Compression Enable		
		Value	Name
		0	Compression Disable
		1	Compression Enable
23	Reference Picture 11 - Compression Type		
		Value	Name
		0	Media Compression Enabled [Default]
		1	Render Compression Enabled
22	Reference Picture 11 - Memory Compression Enable		
		Value	Name
		0	Compression Disable
		1	Compression Enable
21	Reference Picture 10-Compression Type		
		Value	Name
		0	Media Compression Enabled [Default]
		1	Render compression enabled
20	Reference Picture 10 - Memory Compression Enable		
		Value	Name
		0	Compression Disable
		1	Compression Enable
19	Reference Picture 9 - Compression Type		
		Value	Name
		0	Media Compression Enabled [Default]
		1	Render Compression Enabled
18	Reference Picture 9 - Memory Compression Enable		
		Value	Name
		0	Compression Disable
		1	Compression Enable
17	Reference 8 - Compression Type		
		Value	Name

MFX_PIPE_BUF_ADDR_STATE		
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled
16	Reference Picture 8 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
15	Reference Picture 7 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled
14	Reference Picture 7 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
13	Reference Picture 6 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled
12	Reference Picture 6 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
11	Reference Picture 5 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled
10	Reference Picture 5 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
9	Reference Picture 4 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]

MFX_PIPE_BUF_ADDR_STATE

	1	Render Compression Enabled
8	Reference Picture 4 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
7	Reference Picture 3 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled
6	Reference Picture 3 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
5	Reference Picture 2 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled
4	Reference Picture 2 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
3	Reference Picture 1 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled
2	Reference Picture 1 - Memory Compression Enable	
	Value	Name
	0	Compression Disable
	1	Compression Enable
1	Reference Picture 0 - Compression Type	
	Value	Name
	0	Media Compression Enabled [Default]
	1	Render Compression Enabled

MFX_PIPE_BUF_ADDR_STATE				
	0	Reference Picture 0 - Memory Compression Enable		
		Value	Name	
		0	Compression Disable	
		1	Compression Enable	
62	31:6	Scaled Reference Surface Base Address		
		Format: GraphicsAddress[31:6] Specifies the 64 byte aligned down scaled reference frame buffer addresses that needs to be used by the PAK down-scaler to write the down scaled pixels. Only the luma pixels will be downscaled and written to the surface		
	5:0	Reserved		
		Format: MBZ		
63	31:16	Reserved		
		Format: MBZ		
	15:0	Scaled Reference Surface Base Address High		
		Format: GraphicsAddress[47:32] This field is for the upper range of Scaled Reference Surface Base Address.		
64	31:15	Reserved		
		Format: MBZ		
	14:13	Scaled Reference Surface - Tiled Resource Mode		
		For Media Surfaces:This field specifies the tiled resource mode		
		Value	Name	Description
		0h	TRMODE_NONE	No tiled resource
1h		TRMODE_TILEYF	No tiled resource	
2h	TRMODE_TILEYS	No tiled resource		
3h	Reserved			
	12:11	Reserved		
		Format: MBZ		
	10	Scaled Reference Surface - Render Compression Enable		
		Value	Name	
		0	Disable [Default]	
		1	Enable	
	9	Scaled Reference Surface - Memory Compression Enable		
		Format: Enable Memory compression shouldn't be enabled for this surface.		

MFX_PIPE_BUF_ADDR_STATE																	
	8:7	<p>Scale Reference Surface - Arbitration Priority Control This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest Priority</td> </tr> <tr> <td>01b</td> <td>Second Highest Priority</td> </tr> <tr> <td>10b</td> <td>Third Highest Priority</td> </tr> <tr> <td>11b</td> <td>Lowest Priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest Priority	01b	Second Highest Priority	10b	Third Highest Priority	11b	Lowest Priority					
	Value	Name															
	00b	Highest Priority															
	01b	Second Highest Priority															
	10b	Third Highest Priority															
11b	Lowest Priority																
6:0	<p>Scaled Reference Surface - Memory Object Control State Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).</p>																
65	31:6	<p>SliceSize StreamOut Data Destination Base Address Format: GraphicsAddress[31:6] Specifies the 64 byte aligned Slice Size streamout surface address. Here slice sizes are written out. This surface can be used to determine the slice start location.</p>															
	5:0	<p>Reserved Format: MBZ</p>															
66	31:16	<p>Reserved Format: MBZ</p>															
	15:0	<p>SliceSize StreamOut Data Destination Base Address High Format: GraphicsAddress[47:32] This field is for the upper range of Slice Size Streamout Surface Base Address.</p>															
67	31:15	<p>Reserved Format: MBZ</p>															
	14:13	<p>SliceSize StreamOut Data Destination - Tiled Resource Mode For Media Surfaces: This Surface is never tiled.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>No tiled resource</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>No tiled resource</td> </tr> <tr> <td>3h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	No tiled resource	2h	TRMODE_TILEYS	No tiled resource	3h	Reserved	
	Value	Name	Description														
	0h	TRMODE_NONE	No tiled resource														
1h	TRMODE_TILEYF	No tiled resource															
2h	TRMODE_TILEYS	No tiled resource															
3h	Reserved																
12:11	<p>Reserved Format: MBZ</p>																
10	<p>SliceSize StreamOut Data Destination - Memory Compression Mode Format: <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> </tr> </tbody> </table></p>	Value	Name														
Value	Name																

MFX_PIPE_BUF_ADDR_STATE											
	<table border="1"> <tr> <td style="width: 100px;">0</td> <td>Reserved [Default]</td> </tr> </table>	0	Reserved [Default]								
0	Reserved [Default]										
9	<p>SliceSize StreamOut Data Destination - Memory Compression Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Memory compression is never enabled for this surface</p>	Format:	Enable								
Format:	Enable										
8:7	<p>SliceSize StreamOut Data Destination - Arbitration Priority Control</p> <p>This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Highest Priority</td> </tr> <tr> <td>01b</td> <td>Second Highest Priority</td> </tr> <tr> <td>10b</td> <td>Third Highest Priority</td> </tr> <tr> <td>11b</td> <td>Lowest Priority</td> </tr> </tbody> </table>	Value	Name	00b	Highest Priority	01b	Second Highest Priority	10b	Third Highest Priority	11b	Lowest Priority
Value	Name										
00b	Highest Priority										
01b	Second Highest Priority										
10b	Third Highest Priority										
11b	Lowest Priority										
6:0	<p>SliceSize StreamOut Data Destination - Memory Object Control State</p> <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).</p>	Format:	MEMORY_OBJECT_CONTROL_STATE								
Format:	MEMORY_OBJECT_CONTROL_STATE										

MFX_PIPE_MODE_SELECT

MFX_PIPE_MODE_SELECT			
Source:	VideoCS		
Length Bias:	2		
<p>Specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis.</p> <p>The MFX_PIPE_MODE_SELECT command specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis. It also configures the hardware pipeline according to the active encoder/decoder operating mode for encoding/decoding the current picture. Commands issued specifically for AVC and MPEG2 are ignored when VC1 is the active codec.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_COMMON
		Format:	OpCode
	26:24	Opcode	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	SubOpA	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpB		
	Default Value:	0h MFX_PIPE_MODE_SELECT	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Format:	=n	
	Value	Name	Description
	3h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)
1	31:24	AES Control	
		Format:	AES_CONTROL
	23:19	Reserved	
	Format:	MBZ	

MFX_PIPE_MODE_SELECT

18	Extended stream out enable <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">U1</td> </tr> </table> <p>This bit can be set only when VDEnc_Mode is set. When this bit is set and MB stream out is enabled, per MB 1CL of data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder VDEnc mode StreamOut Data StructureDefinition. When this bit is not set, per MB CL data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder StreamOut Mode Data Structure Definition.</p>		Format:	U1													
Format:	U1																
17	Decoder Short Format Mode For IT mode, this bit must be 0. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Short Format Driver Interface [Default]</td> <td>AVC/VC1/MVC/VP8 Short Format Mode is in use Note: There is no Short Format for VP8 yet, so this field must be set to 1 for VP8.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Long Format Driver Interface</td> <td>AVC/VC1/MVC/VP8 Long Format Mode is in use.</td> </tr> </tbody> </table>		Value	Name	Description	0	Short Format Driver Interface [Default]	AVC/VC1/MVC/VP8 Short Format Mode is in use Note: There is no Short Format for VP8 yet, so this field must be set to 1 for VP8.	1	Long Format Driver Interface	AVC/VC1/MVC/VP8 Long Format Mode is in use.						
Value	Name	Description															
0	Short Format Driver Interface [Default]	AVC/VC1/MVC/VP8 Short Format Mode is in use Note: There is no Short Format for VP8 yet, so this field must be set to 1 for VP8.															
1	Long Format Driver Interface	AVC/VC1/MVC/VP8 Long Format Mode is in use.															
16:15	Decoder Mode select Each coding standard supports two entry points: VLD entry point and IT (IDCT) entry point. This field selects which one is in use. This field is only valid if Codec Select is 0 (decoder). <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>VLD Mode</td> <td>All codec minimum must support this mode Configure the MFD Engine for VLD Mode Note: All codec minimum must support this mode</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>IT Mode</td> <td>Configure the MFD Engine for IT Mode Note: Only VC1 and MPEG2 support this mode</td> </tr> <tr> <td style="text-align: center;">2h</td> <td>Deblocker Mode</td> <td>Configure the MFD Engine for Standalone Deblocker Mode. Require streamout AVC edge control information from preceeding decoding pass. Note: Only AVC, MPEG2 are supported.</td> </tr> <tr> <td style="text-align: center;">3h</td> <td>Interlayer Mode</td> <td>Configure the MFX Engine for standalone interlayer upsampling for motion info, residual and reconstructed pixel. Require information being streamout from the preceding encoding and decoding pass of a reference layer.></td> </tr> </tbody> </table>		Value	Name	Description	0h	VLD Mode	All codec minimum must support this mode Configure the MFD Engine for VLD Mode Note: All codec minimum must support this mode	1h	IT Mode	Configure the MFD Engine for IT Mode Note: Only VC1 and MPEG2 support this mode	2h	Deblocker Mode	Configure the MFD Engine for Standalone Deblocker Mode. Require streamout AVC edge control information from preceeding decoding pass. Note: Only AVC, MPEG2 are supported.	3h	Interlayer Mode	Configure the MFX Engine for standalone interlayer upsampling for motion info, residual and reconstructed pixel. Require information being streamout from the preceding encoding and decoding pass of a reference layer.>
Value	Name	Description															
0h	VLD Mode	All codec minimum must support this mode Configure the MFD Engine for VLD Mode Note: All codec minimum must support this mode															
1h	IT Mode	Configure the MFD Engine for IT Mode Note: Only VC1 and MPEG2 support this mode															
2h	Deblocker Mode	Configure the MFD Engine for Standalone Deblocker Mode. Require streamout AVC edge control information from preceeding decoding pass. Note: Only AVC, MPEG2 are supported.															
3h	Interlayer Mode	Configure the MFX Engine for standalone interlayer upsampling for motion info, residual and reconstructed pixel. Require information being streamout from the preceding encoding and decoding pass of a reference layer.>															
14	Standalone VDEnc_Mode Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">Enable</td> </tr> </table> <p>This field indicates to PAK if this is standalone VDEnc mode. This is primarily a validation mode.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>VDEnc+PAK</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>PAK Only</td> </tr> </tbody> </table>		Format:	Enable	Value	Name	0h	VDEnc+PAK	1h	PAK Only							
Format:	Enable																
Value	Name																
0h	VDEnc+PAK																
1h	PAK Only																
13	VDEnc_Mode This field indicates if PAK is working in legacy MBEnc mode or the VDEnc mode.																

MFX_PIPE_MODE_SELECT

		Value	Name	Description
		0h	MBEnc mode	PAK is working in legacy mode
		1h	VDEnc mode	PAK is working in VDEnc mode
12	Deblocker Stream-Out Enable This field indicates if Deblocker information is going to be streamout during VLD decoding. For AVC, it is needed to enable the deblocker streamout as the AVC Disable_DLKFilterIdc is a slice level parameters. Driver needs to determine ahead of time if at least one slice of the current frame/ has deblocker ON.			
		Value	Name	Description
		0h	Disable	Disable streamout of deblocking control information for standalone deblocker operation. It needs other fields to determine one or two deblocking surface streamout (Post Deblocking Output Enable, Pre Deblocking Output Enable, interlayer idc and regular deblock idc).
		1h	Enable	
11	Pic Error/Status Report Enable. This field control whether the error/status reporting is enable or not. 0: Disable 1: Enable In decoder modes: Error reporting is written out once per frame. The Error Report frame ID listed in DW3 along with the VLD/IT error status bits are packed into one cache and written to the "Decoded Picture Error/Status Buffer address" listed in the MFX_PIPE_BUF_ADDR_STATE Command. Note: driver shall program different error buffer addresses between pictures; otherwise, hardware might overwrite previous written data if driver does not read it fast enough. In encoder modes: Not used Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format.			
		Value	Name	
		0h	Disable	
		1h	Enable	
10	Stream-Out Enable This field controls whether the macroblock parameter stream-out is enabled during VLD decoding for transcoding purpose.			
		Value	Name	
		0h	Disable	
		1h	Enable	
	Programming Notes			
	In decoder modes: The Stream-Out feature is added to support transcoding. While decoding the input compressed stream, selected decoded information may be used by the encoder for re-compression. In encoder modes: This feature used to perform dynamic Multipass of PAK for conformance purpose. Also it provides feedback to host (ENC) for future needs. Software can use this bit to disable writing PAK steam data to the streamout buffer for last pass of frame in PAK. Thus, save memory bandwidth.			

MFX_PIPE_MODE_SELECT

9	Post Deblocking Output Enable (PostDeblockOutEnable) This field controls the output write for the reconstructed pixels AFTER the deblocking filter. In MPEG2 decoding mode, if this is enabled, VC1 deblocking filter is used.		
	Value	Name	
	0h	Disable	
	1h	Enable	
8	Pre Deblocking Output Enable (PreDeblockOutEnable) This field controls the output write for the reconstructed pixels BEFORE the deblocking filter.		
	Value	Name	
	0h	Disable	
	1h	Enable	
7	Scaled Surface Enable This field indicates if the scaled surface is enabled. This field enables the 4x HME downscaler of the reconstructed image. Only supported for AVC and VP8 formats.		
	Value	Name	
	0h	Disable	
	1h	Enable	
6	Frame Statistics StreamOut Enable This field controls the frame level statistics streamout from the PAK. Note: This field needs to be always "Enabled" in VD_Enc mode. In case of non-VDEnc mode, this can be used to control the frame statistics output from the PAK.		
	Value	Name	
	0h	Disable	
	1h	Enable	
5	Stitch Mode Exists If: //CodecSel=Encode and StandardSel=AVC		
	Value	Name	Description
	0h	Not in stitch mode	
	1h	In the special stitch mode	This mode can be used for any Codec as long as bitfield conditions are met.
4	Codec Select		
	Value	Name	Description
	0h	Decode	
	1h	Encode	Valid only if StandardSel is AVC, MPEG2)
3:0	Standard Select		
	Value	Name	Description
	0000b	MPEG2	

MFX_PIPE_MODE_SELECT			
		0001b	VC1
		0010b	AVC Covers both AVC and MVC
		0011b	JPEG
		0101b	VP8 Decoder
		0110b	Reserved
		0111b	Reserved
		1111b	UVLD SW decoder w/ embedded micro-controller and co-processor
2	31:0	Reserved	
		Format:	MBZ
3	31:0	Pic Status/Error Report ID	
		Exists If:	//Decoder Mode Only
		Format:	U32
		In decoder modes: Error reporting is written out once per frame. This field along with the VLD error status bits are packed into one cache and written to the memory location specified by "Decoded Picture Error/Status Buffer address" listed in the MFX_PIPE_BUF_ADDR_STATE Command.	
		Value	Name
		0h	32-bit unsigned Unique ID Number
		1h	Reserved
4	31:0	Reserved	
		Format:	MBZ

MFX_QM_STATE

MFX_QM_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This is a common state command for AVC encoder modes. For encoder, it represents both the forward QM matrices as well as the decoding QM matrices. This is a Frame-level state. Only Scaling Lists specified by an application are being sent to the hardware. The driver is responsible for determining the final set of scaling lists to be used for decoding the current slice, based on the AVC Spec Table 7-2 (Fall-Back Rules A and B). In MFX AVC PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order. But the Forward Q scaling lists are sent in column-wise raster order (column-by-column) to simplify the H/W. Driver will perform all the scan order conversion for both Forward Q and IQ.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_MULTI_DW
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	7h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	20h Excludes DWord (0,1)	
	Format:	=n	
1	31:2	Reserved	
		Format:	MBZ
	1:0	AVC	
Exists If:		//AVC- Decoder Only	
For AVC QM Type: This field specifies which Quantizer Matrix is loaded.			

		MFX_QM_STATE											
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td> </tr> <tr> <td>1</td> <td>AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)</td> </tr> <tr> <td>2</td> <td>AVC_8x8_Intra_MATRIX</td> </tr> <tr> <td>3</td> <td>AVC_8x8_Inter_MATRIX</td> </tr> </tbody> </table>	Value	Name	0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)	1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)	2	AVC_8x8_Intra_MATRIX	3	AVC_8x8_Inter_MATRIX	
Value	Name												
0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)												
1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)												
2	AVC_8x8_Intra_MATRIX												
3	AVC_8x8_Inter_MATRIX												
	1:0	<p>MPEG2</p> <p>Exists If: //MPEG2- Decoder Only</p> <p>For MPEG2 QM Type: This field specifies which Quantizer Matrix is loaded.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>MPEG_INTRA_QUANTIZER_MATRIX</td> </tr> <tr> <td>1</td> <td>MPEG_NON_INTRA_QUANTIZER_MATRIX</td> </tr> <tr> <td>2-3</td> <td>Reserved</td> </tr> </tbody> </table>		Value	Name	0	MPEG_INTRA_QUANTIZER_MATRIX	1	MPEG_NON_INTRA_QUANTIZER_MATRIX	2-3	Reserved		
Value	Name												
0	MPEG_INTRA_QUANTIZER_MATRIX												
1	MPEG_NON_INTRA_QUANTIZER_MATRIX												
2-3	Reserved												
	1:0	<p>JPEG</p> <p>Exists If: //JPEG- Encoder Only</p> <p>For JPEG QM Type: This field specifies which Quantizer Matrix is loaded.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>JPEG_Luma_Y_QUANTIZER_MATRIX (or R)</td> </tr> <tr> <td>1</td> <td>JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)</td> </tr> <tr> <td>2</td> <td>JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j]. In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.</p>		Value	Name	0	JPEG_Luma_Y_QUANTIZER_MATRIX (or R)	1	JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)	2	JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)		
Value	Name												
0	JPEG_Luma_Y_QUANTIZER_MATRIX (or R)												
1	JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)												
2	JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)												
2..33	1023:0	<p>Forward Quantizer Matrix</p> <p>The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.</p>											

MFX_STATE_POINTER

MFX_STATE_POINTER			
Source:	VideoCS		
Length Bias:	2		
<p>The MFX_STATE_POINTER command, issued at picture level, issued to set up the indirect pointers for VCS to fetch all the MFX states (Image state, Slice state, etc.) needed for the encoding/decoding process in PAK/IT mode. The encoding/decoding states are presented by state commands, which are grouped into separate sets (picture level, slice level, etc.), and each is stored in its own memory buffer referred by an indirect state pointer. The content of each indirect state buffer is a list of MFX state commands with no special format requirements. The sequence of commands in each indirect state buffer is terminated by a MI_BATCH_BUFFER_END command(acts as the last command marker). Therefore, indirect state buffers can have different and variable length of command sequences.</p> <p>The indirection is designed to facilitate context switching in the middle of a codec operation. The smallest granularity of interruption is designed to be at a completed MB row in AVC/VC1/MPEG2 IT and AVC PAKoperating modes as well as in VC1/MPEG2 VLD mode. There is no support for context switchin AVC VLD mode.</p> <p>Hardware supports up to 4 separate indirect state pointers, allowing software to manage the grouping of state commands. During context switch, hardware restores (re-issues) the latest version of each indirect state pointer, if present.</p> <p>MFX_STATE_POINTER command can only program one indirect state pointer at a time.MI_FLUSH will invalidate all indirect state buffer pointers inside VCS.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFX_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Media
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	SubOpcode A	
		Default Value:	0h
		Format:	OpCode
	20:16	SubOpcode B	
		Default Value:	6h
Format:		OpCode	
15:12	Reserved		
	Format:	MBZ	

MFX_STATE_POINTER																	
	11:0	DWord Length															
		<table border="1"> <tr> <td>Default Value:</td> <td>0h DWORD_COUNT_n</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	0h DWORD_COUNT_n	Format:	=n											
Default Value:	0h DWORD_COUNT_n																
Format:	=n																
1	31:5	State Pointer															
		<table border="1"> <tr> <td>Format:</td> <td>GeneralStateOffset[31:5]</td> </tr> </table> <p>Specifies the 32-byte aligned address of an Indirect State Buffer. This pointer is relative to the General State Base Address.</p>	Format:	GeneralStateOffset[31:5]													
	Format:	GeneralStateOffset[31:5]															
4:2	Reserved																
		<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
Format:	MBZ																
	1:0	State Pointer Index															
		Specifies one of the four indirect state pointers to program.															
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td></td> <td>indirect state pointer 0 (image state)</td> </tr> <tr> <td>01b</td> <td></td> <td>indirect state pointer 1 (slice state)sc</td> </tr> <tr> <td>10b</td> <td></td> <td>indirect state pointer 2</td> </tr> <tr> <td>11b</td> <td></td> <td>indirect state pointer 3</td> </tr> </tbody> </table>	Value	Name	Description	00b		indirect state pointer 0 (image state)	01b		indirect state pointer 1 (slice state)sc	10b		indirect state pointer 2	11b		indirect state pointer 3
Value	Name	Description															
00b		indirect state pointer 0 (image state)															
01b		indirect state pointer 1 (slice state)sc															
10b		indirect state pointer 2															
11b		indirect state pointer 3															

MFX_STITCH_OBJECT

MFX_STITCH_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>The MFC_STITCH_OBJECT command is used when stitch-enabled is set to 1, while CodecSel and StandardSel are set to ENCODE and AVC, respectively. This command is used, for example, to stitch multiple bitstreams to form a transport stream.</p> <p>It is a variable length command as the data to be inserted are presented as either inline data and/or indirect data of this command. Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid output. Hardware keeps track of an output bitstream buffer current byte position and the associated next bit insertion position index. Context switch interrupt is not supported by this command.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFC_STITCH_OBJECT
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	0h MFX_COMMON
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		2h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	Ah	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	[0h, FFFh] Excludes DWord (0,1) = Variable Length in DW (>= 3)	
	Format:	=n	
	If it is 3, it indicates the absent of inline data.		
1	31:18	Reserved	
		Format:	MBZ

MFX_STITCH_OBJECT						
1	17:16	Source Data Starting Byte Offset Source Data Starting Byte Position within the very first inline DW.				
	15:14	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ		
		MBZ				
	13:8	Source Data Ending Bit Inclusion Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first. For example, SrCDataEndingBitInclusion =9, bit 7:0 and bit 15 are included as valid header data. <table border="1" style="width: 100%; text-align: center;"><thead><tr><th style="color: blue;">Value</th><th style="color: blue;">Name</th></tr></thead><tbody><tr><td>[1,32]</td><td> </td></tr></tbody></table>	Value	Name	[1,32]	
	Value	Name				
	[1,32]					
	7:4	Reserved				
	3	Reserved				
	2	Last Source Header Data Insert Command Flag To process a series of consecutive insertion commands, this flag (= 1) indicates the current command is the last 'header' insertion in the series. In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command. In CAVLC, hardware ignores this bit.				
	1	Last Destination Data Insert Command Flag <table border="1" style="width: 100%;"><tr><td>THIS FIELD MUST BE THE SAME AS Last Source Header Data Insert Command Flag</td></tr><tr><td>No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory</td></tr></table>	THIS FIELD MUST BE THE SAME AS Last Source Header Data Insert Command Flag	No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory		
THIS FIELD MUST BE THE SAME AS Last Source Header Data Insert Command Flag						
No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory						
0	Reserved					
2	31:19	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ		
		MBZ				
18:0	Indirect Data Length Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>U19</td></tr></table> This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address.		U19			
	U19					
3	31:0	Indirect Data Start Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>GraphicsAddress[31:0]</td></tr></table> This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the MFX Indirect Bitstream Object Base Address. Hardware ignores this field if indirect data is not present.		GraphicsAddress[31:0]		
		GraphicsAddress[31:0]				
4..n	31:0 Insert Data Payload Inline data to be inserted to the output bitstream buffer					

MFx_SURFACE_STATE

MFx_SURFACE_STATE	
Source:	VideoCS
Length Bias:	2
Description	
<p>This command is common for all encoding/decoding modes, to specify the uncompressed YUV picture (i.e. destination surface) or intermediate streamout in/out surface (e.g. coefficient/residual) (field, frame or interleaved frame) format for reading and writing:</p> <ul style="list-style-type: none"> • Uncompressed, original input picture to be encoded • Reconstructed non-filtered/filtered display picture (becoming reference pictures as well for subsequent temporal inter-prediction) <p>Since there is only one media surface state being active during the entire encoding/decoding process, all the uncompressed/reconstructed pictures are defined to have the same surface state. For each media object call (decoding or encoding), to distinguish among them, a surfaceID is added to specify for each type of surface. The primary difference among picture surface states is their individual programmed base addresses, which are provided by other state commands and not included in this command. MFX engine is making the association of surface states and corresponding buffer base addresses.</p> <p>MFX engine currently supports only one media surface type for video and that is the NV12 (Planar YUV420 with interleaved U (Cb) and V (Cr)). For optimizing memory efficiency based on access patterns, only TileY is supported. For JPEG decoder, only IMC1 and IMC3 are supported. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. The following describes all the different formats that are supported and not supported in MFX :</p> <ul style="list-style-type: none"> • NV12 - 4:2:0 only; UV interleaved; Full Pitch, U and V offset is set to 0 (the only format supported for video codec); vertical UV offset is MB aligned; UV xoffsets = 0. JPEG does not support NV12 format because non-interleave JPEG has performance issue with partial write (interleaved UV format) • IMC 1 & 3 - Full Pitch, U and V are separate plane; (JPEG only; U plane + garbage first in full pitch followed by V plane + garbage in full pitch). U and V vertical offsets are block aligned; U and V xoffset = 0; there is no gap between Y, U and V planes. IMC1 and IMC3 are different by a swap of U and V. This is the only format supported in JPEG for all video subsampling types (4:4:4, 4:2:2 and 4:2:0) • We are not supporting IMC 2 & 4 - Full Pitch, U and V are separate plane (JPEG only; U plane first in full pitch followed by V plane in full pitch - U and V plane are side-by-side). U and V vertical offsets are 16-pixel aligned; V xoffset is half-pitch aligned; U xoffset is 0; there is no gap between Y, U and V planes. IMC2 and IMC4 are different by a swap of U and V. • We are not supporting YV12 - half pitch for each U and V plane, and separate planes for Y, U and V (U plane first in half pitch followed by V plane in half pitch). For YV12, U and V vertical offsets are block aligned; U and V xoffset = 0; there is no gap between Y, U and V planes <p>Note that the following data structures are not specified through the media surface state</p> <ul style="list-style-type: none"> • 1D buffers for row-store and other miscellaneous information. • 2D buffers for per-MB data-structures (e.g. DMV buffer, MB info record, ILDB Control and Tcoeff/Stcoeff). 	

MFX_SURFACE_STATE

This surface state here is identical to the Surface State for deinterlace and sample_8x8messages described in the Shared Function Volume.

For non pixel data, such as row stores, indirect data (Compressed Slice Data, AVC MV record, Coeff record and AVC ILDB record) and streamin/out and output compressed bitstream, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This mechanism is chosen over the pixelsurface type because of their variable record sizes.

All row store surfaces are linear surface. Their addresses are programmed in Pipe_Buf_Base_State or Bsp_Buf_Base_Addr_State

This surface state here is identical to the Surface State for deinterlace and sample_8x8messages described in the Shared Function Volume and Sampler Chapter.

Programming Notes

VC1 I picture scaling: Even though VC1 allows I reconstructed picture scaling (via RESPIC), as such scaling is only allowed at I picture. All subsequent P (and B) pictures must have the same picture dimensions with the preceding I picture. Therefore, all reference pictures for P or B picture can share the same surface state with the current P and B picture. Note : H/W is not processing RESPIC. Application is no longer expecting intel decoder pipeline and kernel to perform this function, it is going to be done in the video post-processing scaler or display controller scale as a separate step and controller.

All video codec surfaces must be NV12 Compliant, except JPEG. U/V vertical must be MB aligned for all video codec (further constrained for field picture), but JPEG can be block aligned. All video codec and JPEG uses Tiled - Y format only, for uncompressed pixel surfaces.

Even for JPEG planar 420 surface, application may provide only 1 buffers, but there is still only one single surface state for all of them. If IMC equal to 1, 2, 3 or 4, U and V have the pitch same as Y. And U and V will have different offset, each offset is block aligned.

DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27		Pipeline	
			Default Value:	2h MFX_COMMON
			Format:	OpCode
	26:24		Opcode	
			Default Value:	0h MFX_COMMON_STATE
			Format:	OpCode
	23:21		SubOpA	
			Default Value:	0h
			Format:	OpCode
20:16		SubOpB		

MFx_SURFACE_STATE																
		<table border="1"> <tr> <td>Default Value:</td> <td>1h</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	1h	Format:	OpCode										
Default Value:	1h															
Format:	OpCode															
	15:12	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ												
Format:	MBZ															
	11:0	DWord Length <table border="1"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>4h</td> <td>DWORD_COUNT_n [Default]</td> <td>Excludes DWord (0,1)</td> </tr> </tbody> </table>	Format:	=n	Value	Name	Description	4h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)						
Format:	=n															
Value	Name	Description														
4h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)														
1	31:4	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ												
Format:	MBZ															
	3:0	Surface Id <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> </table> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0100b</td> <td>Source Input Picture (encoder)</td> <td>8-bit uncompressed data</td> </tr> <tr> <td>0101b</td> <td>Reconstructed Scaled Reference Picture</td> <td>8-bit data</td> </tr> </tbody> </table>	Format:	U4	Value	Name	Description	0100b	Source Input Picture (encoder)	8-bit uncompressed data	0101b	Reconstructed Scaled Reference Picture	8-bit data			
Format:	U4															
Value	Name	Description														
0100b	Source Input Picture (encoder)	8-bit uncompressed data														
0101b	Reconstructed Scaled Reference Picture	8-bit data														
2	31:18	Height <table border="1"> <tr> <td>Format:</td> <td>U14-1</td> </tr> </table> <p>This field specifies the height of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. Note : Video Codecs must program less than and equal to 4K.(In future, it will be ideal to have this field define in a WORD boundary.)AVC - multiple of 2 MB rows for field picture VC1 - multiple of 4 pixels for field pictureMPEG2 - multiple of 2 MB rows for field pic JPEG - multiple of integral MCU (8 or 16 pixels) per picture</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[0,16383]</td> <td></td> <td>representing heights [1,16384]</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="3" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="3"> <ul style="list-style-type: none"> For AVC : For frame picture is a multiple of 16; for field picture is a multiple of 32 For VC1 : For progressive frames, the frame height and frame width is a multiple of 2 pixels. For interlaced frames, the frame height shall be a multiple of 4 pixels, and its width is a multiple of 2 pixels, based on a PLANAR_420 surface. </td> </tr> </tbody> </table>	Format:	U14-1	Value	Name	Description	[0,16383]		representing heights [1,16384]	Programming Notes			<ul style="list-style-type: none"> For AVC : For frame picture is a multiple of 16; for field picture is a multiple of 32 For VC1 : For progressive frames, the frame height and frame width is a multiple of 2 pixels. For interlaced frames, the frame height shall be a multiple of 4 pixels, and its width is a multiple of 2 pixels, based on a PLANAR_420 surface. 		
Format:	U14-1															
Value	Name	Description														
[0,16383]		representing heights [1,16384]														
Programming Notes																
<ul style="list-style-type: none"> For AVC : For frame picture is a multiple of 16; for field picture is a multiple of 32 For VC1 : For progressive frames, the frame height and frame width is a multiple of 2 pixels. For interlaced frames, the frame height shall be a multiple of 4 pixels, and its width is a multiple of 2 pixels, based on a PLANAR_420 surface. 																
	17:4	Width <table border="1"> <tr> <td>Format:</td> <td>U14-1</td> </tr> </table> <p>This field specifies the width of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the width of the Y (luma) plane.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> </tbody> </table>	Format:	U14-1	Value	Name	Description									
Format:	U14-1															
Value	Name	Description														

MFX_SURFACE_STATE																																													
		<table border="1"> <tr> <td>[0,16383]</td> <td></td> <td>representing widths [1,16384]</td> </tr> <tr> <td colspan="3" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="3"> <ul style="list-style-type: none"> The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). Width (field value + 1) must be a multiple of 2 for PLANAR_420, MFX HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT. </td> </tr> </table>	[0,16383]		representing widths [1,16384]	Programming Notes			<ul style="list-style-type: none"> The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). Width (field value + 1) must be a multiple of 2 for PLANAR_420, MFX HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT. 																																				
[0,16383]		representing widths [1,16384]																																											
Programming Notes																																													
<ul style="list-style-type: none"> The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). Width (field value + 1) must be a multiple of 2 for PLANAR_420, MFX HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT. 																																													
3:2	Reserved	<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																																									
Format:	MBZ																																												
1:0	Cr(V)/Cb(U) Pixel Offset V Direction	<table border="1"> <tr> <td>Format:</td> <td>U0.2</td> </tr> </table> <p>Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction</p> <table border="1"> <tr> <td colspan="3" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="3">This field is ignored for all formats except PLANAR_420_8</td> </tr> </table>	Format:	U0.2	Programming Notes			This field is ignored for all formats except PLANAR_420_8																																					
Format:	U0.2																																												
Programming Notes																																													
This field is ignored for all formats except PLANAR_420_8																																													
3	31:28	<table border="1"> <tr> <td colspan="2">Surface Format</td> </tr> <tr> <td>Format:</td> <td>U4</td> </tr> </table> <p>Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1. Usage: For 420 planar YUV surface, use 4; for monochrome surfaces, use 12. For monochrome surfaces, hardware ignores control fields for Chroma planes. This field must be set to 4 - PLANAR_420_8, or 12 - Y8_UNORM. Not used for MFX, and is ignored. But for JPEG decoding, this field should be programmed to the same format as JPEG_PIC_STATE. For video codec, it should set to 4 always.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>YCRCB_NORMAL</td> <td></td> </tr> <tr> <td>1</td> <td>YCRCB_SWAPUVY</td> <td></td> </tr> <tr> <td>2</td> <td>YCRCB_SWAPUV</td> <td></td> </tr> <tr> <td>3</td> <td>YCRCB_SWAPY</td> <td></td> </tr> <tr> <td>4</td> <td>PLANAR_420_8</td> <td>(NV12, IMC1,2,3,4, YV12)</td> </tr> <tr> <td>5</td> <td>PLANAR_411_8</td> <td>Deinterlace Only</td> </tr> <tr> <td>6</td> <td>PLANAR_422_8</td> <td>Deinterlace Only</td> </tr> <tr> <td>7</td> <td>STMM_DN_STATISTICS</td> <td>Deinterlace Only</td> </tr> <tr> <td>8</td> <td>R10G10B10A2_UNORM</td> <td>Sample_8x8 Only</td> </tr> <tr> <td>9</td> <td>R8G8B8A8_UNORM</td> <td>Sample_8x8 Only</td> </tr> <tr> <td>10</td> <td>R8B8_UNORM (CrCb)</td> <td>Sample_8x8 Only</td> </tr> <tr> <td>11</td> <td>R8_UNORM (Cr/Cb)</td> <td>Sample_8x8 Only</td> </tr> </tbody> </table>	Surface Format		Format:	U4	Value	Name	Description	0	YCRCB_NORMAL		1	YCRCB_SWAPUVY		2	YCRCB_SWAPUV		3	YCRCB_SWAPY		4	PLANAR_420_8	(NV12, IMC1,2,3,4, YV12)	5	PLANAR_411_8	Deinterlace Only	6	PLANAR_422_8	Deinterlace Only	7	STMM_DN_STATISTICS	Deinterlace Only	8	R10G10B10A2_UNORM	Sample_8x8 Only	9	R8G8B8A8_UNORM	Sample_8x8 Only	10	R8B8_UNORM (CrCb)	Sample_8x8 Only	11	R8_UNORM (Cr/Cb)	Sample_8x8 Only
Surface Format																																													
Format:	U4																																												
Value	Name	Description																																											
0	YCRCB_NORMAL																																												
1	YCRCB_SWAPUVY																																												
2	YCRCB_SWAPUV																																												
3	YCRCB_SWAPY																																												
4	PLANAR_420_8	(NV12, IMC1,2,3,4, YV12)																																											
5	PLANAR_411_8	Deinterlace Only																																											
6	PLANAR_422_8	Deinterlace Only																																											
7	STMM_DN_STATISTICS	Deinterlace Only																																											
8	R10G10B10A2_UNORM	Sample_8x8 Only																																											
9	R8G8B8A8_UNORM	Sample_8x8 Only																																											
10	R8B8_UNORM (CrCb)	Sample_8x8 Only																																											
11	R8_UNORM (Cr/Cb)	Sample_8x8 Only																																											

MFX_SURFACE_STATE

	12	Y8_UNORM	Sample_8x8 Only	
27	Interleave Chroma			
	Format:		Enable	
	<p>This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats. For AVC/VC1/MPEG VLD and IT modes : set to Enable to support interleave U/V only. For JPEG : set to Disable for all formats (including 4:2:0) - because JPEG does not support NV12. (This field is needed only if JPEG will support NV12; otherwise is ignored.)</p>			
	Value	Name		
	1	Enable		
	0	Disable		
26:20	Reserved			
	Format:		MBZ	
19:3	Surface Pitch			
	Format:		U17-1	
	<p>This field specifies the surface pitch in (#Bytes).</p>			
	Value	Name		
	[0,131071]			
	Programming Notes			
	<p>For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]</p>			
	<p>For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case). The range in bytes is $[2^{Cu}-1, 131071]$ -> $[(2^{Cu})B, 128KB]$ = [1 tile, 128KB/(2^{Cu}) tiles]</p> <p>The field specifies the surface pitch in (#Bytes - 1)</p>			
	<p>If Media Memory Compression is enabled, the following max pitch size restriction must be honored. For larger resolution, Media Memory compression Must be disabled.</p>			
	Tiling Mode	Pixel Format	Max Frame Width (bytes)	Max Frame Width (pixels)
	Legacy 4K	8bpp	16k	16k
		16bpp	16k	8k
		32bpp	16k	4k
		64bpp	16k	2k
		128bpp	16k	1k
	TileYF	8bpp	8k	8k
		16bpp	16k	8k
			Max Pitch (bytes)	
			16k + 127	
			16k + 127	
			16k + 127	
			16k + 127	
			8k + 63	
			16k + 127	

MFX_SURFACE_STATE

MFX_SURFACE_STATE				
		32bpp	16k	4k
		64bpp	16k	2k
		128bpp	16k	1k
	TileYS	8bpp	16k	16k
		16bpp	16k	8k
		32bpp	16k	4k
		64bpp	16k	2k
		128bpp	16k	1k
				16k + 127
				16k + 255
				16k + 255
				16k + 255
				16k + 511
				16k + 511
				16k + 1023
				16k + 1023
2	Half Pitch for Chroma			
	Format:		Enable	
	(This field must be set to Disable)This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats. This field is ignored by MFX (unless we support YV12)			
1	Tiled Surface			
	Format:		Boolean	
	(This field must be set to TRUE: Tiled)This field specifies whether the surface is tiled. This field is ignored by MFX			
	Value	Name	Description	
	0	False	Linear	
	1	True	Tiled	
	Programming Notes			
	Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory. The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.			
0	Tile Walk			
	Format:		Boolean	
	(This field must be set to 1: TILEWALK_YMAJOR)This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See Memory Interface Functions for details on memory tiling and restrictions. This field is ignored when the surface is linear. This field is ignored by MFX. Internally H/W is always treated this set to 1 for all video codec and for JPEG.			
	Value	Name	Description	
	0h	XMAJOR	TILEWALK_XMAJOR	
	1h	YMAJOR	TILEWALK_YMAJOR	
	Programming Notes			
	The corresponding cache(s) must be invalidated before a previously accessed surface is			

MFx_SURFACE_STATE		
		accessed again with an altered state of this bit
4	31	Reserved Format: _____ MBZ
	30:16	X Offset for U(Cb) Format: _____ U15 This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats. This field must be set to zero. X Offset for U(Cb) in pixel (This field must be zero for NV12 and IMC 1 and 3) <div style="text-align: center;">Programming Notes</div> For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.
	15	Reserved Format: _____ MBZ
	14:0	Y Offset for U(Cb) Format: _____ U15 This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats. <div style="text-align: center;">Programming Notes</div> For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.
	31:29	Reserved Format: _____ MBZ
5	28:16	X Offset for V(Cr) Format: _____ U13 This field must be zero for NV12 and IMC 1 and 3 This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. <div style="text-align: center;">Programming Notes</div> For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:0	Y Offset for V(Cr) Format: _____ U16 This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. This field is ignored by all video codec, only used by JPEG.

MFX_SURFACE_STATE		
		Programming Notes
		For PLANAR_420 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_DIRECTMODE_STATE			
Source:	VideoCS		
Length Bias:	2		
Exists If:	//VC1 decoding in VLD modes		
<p>This is a picture level command and should be issued only once, even for a multi-slices picture. There is only one DMV buffer for read (when processing a B-picture) and one for write (when processing a P-Picture). Each DMV record is 64 bits per MB, to store the top and bottom field MVs (32-bit MV_{x,y} each).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_VC1_DIRECTMODE_STATE
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	2h VC1_COMMON
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	2h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	0005h Excludes DWord (0,1)	
	Format:	=n	
1..2	63:0	Direct MV Write Buffer - Base Address	
		Format:	SplitBaseAddress64ByteAligned
<p>This field provides the base address of the DMV write buffer to store the motion vectors decoded in the current picture. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. This buffer must be 64-byte cacheline aligned. The write buffer size is 557,056 bytes for 1 frame. Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution). This field is only valid for a P picture</p>			

MFX_VC1_DIRECTMODE_STATE		
3	31:0	Direct MV Write Buffer - Attributes Format: MemoryAddressAttributes
4..5	63:0	Direct MV Reference Buffer - Base Address Format: SplitBaseAddress64ByteAligned This field provides the base address of the DMV buffer for reference picture. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned. This field is only valid for a B picture.
6	31:0	Direct MV Reference Buffer - Attributes Format: MemoryAddressAttributes

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_PRED_PIPE_STATE		
Source:	VideoCS	
Length Bias:	2	
<p>This command is used to set the operating states of the MFD Engine beyond the BSD unit. It is used with both VC1 Long and Short format. Driver is responsible to take the intensity compensation enable signal, the LumScale and the LumShift provided from the VC1 interface, and maintain a history of these values for reference pictures. Together with these three parameters specified for the current picture being decoded, driver will derive and supply the above sets of LumScaleX, LumShiftX and intensity compensation enable (single or double, forward or backward) signals. H/W is responsible to take these state values, and use them to build the lookup table (including the derivation of iScale and iShift) for remapping the reference frame pixels, as well as performing the actual pixel remapping calculations/process.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h PARALLEL_VIDEO_PIPE
	Format: OpCode	
	28:27	Pipeline
		Default Value: 2h MFX_VC1_PRED_PIPE_STATE
	Format: OpCode	
	26:24	Media Command Opcode
		Default Value: 2h VC1_COMMON
	Format: OpCode	
	23:21	SubOpcode A
Default Value: 0h		
Format: OpCode		
20:16	SubOpcode B	
	Default Value: 1h	
Format: OpCode		
15:12	Reserved	
Format: MBZ		
11:0	DWord Length	
	Default Value: 0004h Excludes DWord (0,1)	
	Format: =n	
1	31:16	Reserved
		Format: MBZ
15:14	vin_intensitycomp_Double_FWDen	Format: U2

MFV_VC1_PRED_PIPE_STATE			
	<p>for forward reference picture only, to enable top or/and bottom of the reference field enable for single compensation. For frame, may only need one bit. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.</p>		
13:12	<p>vin_intensitycomp_Double_BWDen</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>for backward reference picture only, no double for backward reference. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.</p>	Format:	U2
Format:	U2		
11:10	<p>vin_intensitycomp_Single_FWDen</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>for forward reference picture only, to enable top or/and bottom of the reference field enable for single compensation. For frame, may only need one bit. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.</p>	Format:	U2
Format:	U2		
9:8	<p>vin_intensitycomp_Single_BWDen</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U2</td> </tr> </table> <p>for backward reference picture only, no double for backward reference. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.</p>	Format:	U2
Format:	U2		
7:4	<p>Reference Frame Boundary Replication Mode</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U4</td> </tr> </table> <p>This is a bit field with each bit indicating the corresponding picture's boundary replication mode. Bit 11: reference 3Bit 10: reference 2Bit 9: reference 1Bit 8: reference 00 = progressive frame replication1 = interlace frame replication. This field is maintained and provided by driver for both long and short VC1 interface format.</p>	Format:	U4
Format:	U4		
3:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
2	<p>31:30 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ	
<p>29:24 LumShift2- single - FWD</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U6</td> </tr> </table> <p>This field is maintained and provided by driver for both long and short VC1 interface format.</p>	Format:	U6	
Format:	U6		

MFV_VC1_PRED_PIPE_STATE		
		And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	Reserved
		Format: MBZ
	21:16	LumShift1 - single - FWD
		Format: U6
		This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	Reserved
		Format: MBZ
	13:8	LumScale2 - single - FWD
		Format: U6
		This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	Reserved
		Format: MBZ
	5:0	LumScale1 - Single - FWD
		Format: U6
		This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
3	31:30	Reserved
		Format: MBZ
	29:24	LumShift2- double - FWD
		Format: U6
		This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	Reserved

MFX_VC1_PRED_PIPE_STATE		
		Format: MBZ
	21:16	LumShift1 - double - FWD Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	Reserved Format: MBZ
	13:8	LumScale2 - double - FWD Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	Reserved Format: MBZ
	5:0	LumScale1 - double - FWD Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
4	31:30	Reserved Format: MBZ
	29:24	LumShift2- single - BWD Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	Reserved Format: MBZ
	21:16	LumShift1 - single - BWD Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and

MFV_VC1_PRED_PIPE_STATE	
	wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
15:14	Reserved Format: _____ MBZ
13:8	LumScale2 - single - BWD Format: _____ U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
7:6	Reserved Format: _____ MBZ
5:0	LumScale1 - Single - BWD Format: _____ U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
5	31:30 Reserved Format: _____ MBZ
	29:24 LumShift2- double - BWD Format: _____ U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22 Reserved Format: _____ MBZ
	21:16 LumShift1 - double -BWD Format: _____ U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14 Reserved Format: _____ MBZ

MFV_VC1_PRED_PIPE_STATE			
13:8	<p>LumScale2 - double - BWD</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> <p>This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.</p>	Format:	U6
Format:	U6		
7:6	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
5:0	<p>LumScale1 - double - BWD</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U6</td> </tr> </table> <p>This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCElement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.</p>	Format:	U6
Format:	U6		

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

MFX_VP8_BSP_BUF_BASE_ADDR_STATE			
Source:		VideoCS	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h Video Codec
		Format:	OpCode
	26:24	Media Command OpCode	
		Default Value:	4h VP8
		Format:	OpCode
	23:21	Sub Opcode A	
Default Value:		2h VP8 Common	
Format:		OpCode	
20:16	Sub Opcode B		
	Default Value:	3h MFX_VP8_BSP_BUF_BASE_ADDR_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Format:	=n	
	Value	Name	Description
	000h	Excludes DWord (0,1) [Default]	A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware."
008h		Used for normal encode mode	
1..2	63:0	Frame Header - Base Address	
		Format:	SplitBaseAddress64ByteAligned
		64 byte aligned, 48-bit Abs. Address StreamIn Surface	
		Note: The format is linear vs. tile for better performance.	
3	31:0	Frame Header - Attributes	

MFX_VP8_BSP_BUF_BASE_ADDR_STATE		
		Format: MemoryAddressAttributes
4..5	63:0	Intermediate Buffer - Base Address Format: SplitBaseAddress64ByteAligned 64 byte aligned, 48-bit AbsAddr StreamIn Surface Note: The format is linear vs. tile for better performance.
6	31:0	Intermediate Buffer - Attributes Format: MemoryAddressAttributes
7..14	255:0	Intermediate Buffer Partition Offset <div style="text-align: center; background-color: #e6f2ff; padding: 2px;">Programming Notes</div> All Intermediate Buffer Partition-[i] Offset (i = 1 to 8) and Intermediate Buffer Max Size need to be cacheline aligned (64Byte aligned).
15	31:0	Intermediate Buffer Max Size Format: U32
16..17	63:0	Final Frame - Base Address Format: SplitBaseAddress64ByteAligned 64 byte aligned, 48-bit AbsAddr StreamIn Surface Note: The format is linear vs. tile for better performance.
18	31:0	Final Frame - Attributes Format: MemoryAddressAttributes
19	31:6	Reserved Format: MBZ
	5:0	Final Frame Byte Offset Format: U6 Specify byte offset within a 64-byte cacheline where the bitstream should be inserted at.
20..21	63:0	Streamout - Base Address Format: SplitBaseAddress64ByteAligned 64 byte aligned,48-bit AbsAddr StreamIn Surface Note: The format is linear vs. tile for better performance.
22	31:0	Streamout - Attributes Format: MemoryAddressAttributes
23..24	63:0	Coeff Probs StreamIn Surface - Base Address Format: SplitBaseAddress64ByteAligned

MFx_VP8_BSP_BUF_BASE_ADDR_STATE		
		64 byte aligned, 48-bit AbsAddr StreamIn Surface Note: The format is linear vs. tile for better performance.
25	31:0	Coeff Probs StreamIn Surface - Attributes Format: MemoryAddressAttributes
26..27	63:0	Token Statistics Surface - Base Address Format: SplitBaseAddress64ByteAligned 64 byte aligned, 48-bit Abs. Address StreamIn Surface Note: The format is linear vs. tile for better performance.
28	31:0	Token Statistics Surface - Attributes Format: MemoryAddressAttributes
29..30	63:0	MPC RowStore Surface - Base Address Format: SplitBaseAddress64ByteAligned Abs. Address StreamIn/StreamOut Surface. Note: The format is linear vs. tile for better performance. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.
31	31:0	MPC RowStore Surface - Attributes Format: MemoryAddressAttributes

MFX_VP8_Encoder_CFG

MFX_VP8_Encoder_CFG			
Source:	VideoCS		
Length Bias:	2		
This must be the very first command to issue after the surface state, the pipe select and base address setting commands and must be issued before MFX_VP8_PIC_STATE.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value: 3h PARALLEL_VIDEO_PIPE	
		Format: OpCode	
	28:27	Pipeline	
		Default Value: 2h Video Codec	
	26:24	Media Command OpCode	
		Default Value: 4h VP8	
	23:21	Sub Opcode A	
		Default Value: 2h VP8 Common	
	20:16	Sub Opcode B	
Default Value: 1h MFX_VP8_ENCODER_CFG			
15:12	Reserved		
	Format: MBZ		
11:0	DWord Length		
	Format: =n		
	Value	Name	Description
	000h	Excludes DWord (0,1) [Default]	A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware."
01Dh		Used for normal encode mode	
1	31:11	Reserved	
		Format: MBZ	
1	10	VBPunitPowerClock Gating Disable	
		Format: U1	

MFX_VP8_Encoder_CFG									
	VBSPunit Power Clock Gating Disable.								
9	<p>Compressed Bitstream Output Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Disable Compressed Bitstream Output. (Both Final Bitstream and Intermediate bit buffer)</p>	Format:	U1						
Format:	U1								
8	<p>Finer BRC Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Enable Finer BRC Feature.</p>	Format:	U1						
Format:	U1								
7	<p>Per Segment Delta Qindex / LoopFilter Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Disable Per Segment Delta Qindex / Loop Filter in Rate Control.</p>	Format:	U1						
Format:	U1								
6	<p>Rate Control Initial Pass</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Initial pass</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Subsequence Pass(es)</td> </tr> </tbody> </table>	Format:	U1	Value	Name	1	Initial pass	0	Subsequence Pass(es)
Format:	U1								
Value	Name								
1	Initial pass								
0	Subsequence Pass(es)								
5	<p>Skip Final Bitstream when Over / Under flow</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Skip Final Bitstream conditionally on Over/Under flow in rate control and intermediate Bit Buffer Overrun.</p>	Format:	U1						
Format:	U1								
4	<p>Update Segment Feature Data Flag</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Exists If:</td> <td style="width: 50%;">//VP8 Encoder</td> </tr> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>Enable for Frame Header per Segment Quantizer / LoopFilter Update</p>	Exists If:	//VP8 Encoder	Format:	U1				
Exists If:	//VP8 Encoder								
Format:	U1								
3	<p>Bitstream Statistics Output Enable</p> <p>Enable Bitstream Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]</p>								
2	<p>Token Statistics Output Enable</p> <p>Enable Token Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]</p>								
1	<p>Final Bitstream Output Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>Disable Final Bitstream Output.</p>	Format:	U1						
Format:	U1								

MFX_VP8_Encoder_CFG								
	0	Performance Counter Enable Format: U1 Enable Performance Counter in Streamout.						
		Reserved Format: MBZ						
2	7	Qindex Clamp High mask for overflow Format: U1 If current frame is overflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit1. In another word, subsequent passes would be skipped.						
	6	Qindex Clamp High mask for underflow Format: U1 If current frame is underflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit0. In another word, subsequent passes would be skipped						
	5	Final Bistream Buffer Overrun Enable Mask Format: U1 Enable Final Bitstream Buffer Overrun detection feature.						
	4	Intermediate Bit Buffer Overrun Enable Mask Format: U1 Enable Intermediate Bit Buffer Overrun detection feature.						
	3	Max Intra MB Bit Count Check Enable Mask Format: U1 Enable Max. Intra MB bit count check in Streamout.						
	2	Max Inter MB Bit Count Check Enable Mask Format: U1 Enable Max. Inter MB bit count check in Streamout.						
	1	Min Frame Bit Count Rate Control Enable Mask Format: U1 Enable Min. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is less than or equal to FrameBitRateMin.						
			<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td></td> <td>If (Total Frame Level Bit Counter) = < (Frame Bit Rate Minimum limit) Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS Control</td> </tr> </tbody> </table>	Value	Name	Description	1	
Value	Name	Description						
1		If (Total Frame Level Bit Counter) = < (Frame Bit Rate Minimum limit) Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS Control						

MFX_VP8_Encoder_CFG													
		<table border="1"> <tr> <td></td> <td>Register.</td> </tr> <tr> <td>0</td> <td>Do not update bit[0] of MFX_VP8_IMAGE_STATUS Control Register.</td> </tr> </table>		Register.	0	Do not update bit[0] of MFX_VP8_IMAGE_STATUS Control Register.							
		Register.											
	0	Do not update bit[0] of MFX_VP8_IMAGE_STATUS Control Register.											
	0	<p>Max Frame bit count Rate Control Enable Mask</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>Enable Max. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is greater than or equal to FrameBitRateMax.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td>If (Total Frame Level Bit Counter) >= (Frame Bit Rate Maximum Limit)Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS control register.</td> </tr> <tr> <td>0</td> <td></td> <td>Do not update bit[0] of MFX_VP8_IMAGE_STATUS control register.</td> </tr> </tbody> </table>	Format:	U1	Value	Name	Description	1		If (Total Frame Level Bit Counter) >= (Frame Bit Rate Maximum Limit)Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS control register.	0		Do not update bit[0] of MFX_VP8_IMAGE_STATUS control register.
Format:	U1												
Value	Name	Description											
1		If (Total Frame Level Bit Counter) >= (Frame Bit Rate Maximum Limit)Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS control register.											
0		Do not update bit[0] of MFX_VP8_IMAGE_STATUS control register.											
3	31:28	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	27:16	<p>Max Intra MB Bit Count Limit</p> <table border="1"> <tr> <td>Format:</td> <td>U12</td> </tr> </table> <p>12-bit bit count for Max Intra MB Limit.</p>	Format:	U12									
Format:	U12												
	15:12	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	11:0	<p>Max Inter MB bit count</p> <table border="1"> <tr> <td>Format:</td> <td>U12</td> </tr> </table> <p>12-bit bit count for Max Inter MB Limit.</p>	Format:	U12									
Format:	U12												
4	31	<p>Frame Bitrate Min Unit Mode</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>This field is the Frame Bitrate Minimum Limit Units.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Compatibility Mode</td> <td>Frame BitRate Min Unit is in old mode (128b/16Kb)</td> </tr> <tr> <td>1h</td> <td>New Mode</td> <td>Frame BitRate Min Unit is in new mode (32byte/4Kb)</td> </tr> </tbody> </table>	Format:	U1	Value	Name	Description	0h	Compatibility Mode	Frame BitRate Min Unit is in old mode (128b/16Kb)	1h	New Mode	Frame BitRate Min Unit is in new mode (32byte/4Kb)
		Format:	U1										
		Value	Name	Description									
	0h	Compatibility Mode	Frame BitRate Min Unit is in old mode (128b/16Kb)										
1h	New Mode	Frame BitRate Min Unit is in new mode (32byte/4Kb)											
30	<p>Frame Bit Rate Min Unit</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p><i>This field is Frame Bitrate Minimum Mode.</i></p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>32-B</td> </tr> <tr> <td>1</td> <td>4-KB</td> </tr> </tbody> </table>	Format:	U1	Value	Name	0	32-B	1	4-KB				
Format:	U1												
Value	Name												
0	32-B												
1	4-KB												
29:16	<p>Frame Bit Rate Min</p>												

MFX_VP8_Encoder_CFG

		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U14</td> </tr> </table> <p>If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values</p>	Format:	U14									
Format:	U14												
	15	<p>Frame Bitrate Max Unit Mode</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>This field is the Frame Bitrate Maximum Limit Units.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 15%;">Value</th> <th style="width: 25%;">Name</th> <th style="width: 60%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Compatibility Mode</td> <td>Frame BitRate Max Unit is in old mode (128b/16Kb)</td> </tr> <tr> <td>1h</td> <td>New Mode</td> <td>Frame BitRate Max Unit is in new mode (32byte/4Kb)</td> </tr> </tbody> </table>	Format:	U1	Value	Name	Description	0h	Compatibility Mode	Frame BitRate Max Unit is in old mode (128b/16Kb)	1h	New Mode	Frame BitRate Max Unit is in new mode (32byte/4Kb)
Format:	U1												
Value	Name	Description											
0h	Compatibility Mode	Frame BitRate Max Unit is in old mode (128b/16Kb)											
1h	New Mode	Frame BitRate Max Unit is in new mode (32byte/4Kb)											
	14	<p>Frame Bit Rate Max Unit</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p><i>This field is Frame Bitrate Maximum Mode</i></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 40%;">Value</th> <th style="width: 60%;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>32-B</td> </tr> <tr> <td>1</td> <td>4-KB</td> </tr> </tbody> </table>	Format:	U1	Value	Name	0	32-B	1	4-KB			
Format:	U1												
Value	Name												
0	32-B												
1	4-KB												
	13:0	<p>Frame Bit Rate Max</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U14</td> </tr> </table> <p>If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values</p>	Format:	U14									
Format:	U14												
5	31:24	<p>Frame Delta QIndex Max[3]</p> <p>This field is the Frame level delta Qindex for total bit-count above FrameBitRateMax - First 1/8 Region.</p> <p>This field is used to calculate the suggested Frame Qindex into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax; i.e., In the range of (FrameBitRateMax, (FrameBitRateMax + FrameBitRateMaxDelta » 3)].</p>											
	23:16	<p>Frame DeltaQ Index Max[2]</p> <p>This field is the Frame level delta Qindex for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4.</p> <p>This field is used to calculate the suggested Frame Qindex into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above FrameBitRateMax; i.e., In the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3), (FrameBitRateMax+ FrameBitRateMaxDelta » 2)].</p>											
	15:8	<p>Frame Delta QIndex Max[1]</p> <p>This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/4 and Below 1/2.</p>											

MFX_VP8_Encoder_CFG		
		<p>This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/4 and 1/2 of FrameBitRateMaxDelta above FrameBitRateMax; i.e., In the range of $[(\text{FrameBitRateMax} + \text{FrameBitRateMaxDelta} \gg 2), (\text{FrameBitRateMax} + \text{FrameBitRateMaxDelta} \gg 1)]$.</p>
	7:0	<p>Frame Delta QIndex Max [0] This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMax; i.e., In the range of $[(\text{FrameBitRateMax} + \text{FrameBitRateMaxDelta} \gg 1), (\text{Infinite})]$.</p>
6	31:24	<p>Frame Delta QIndex Min[3] This field is the Frame level delta QINDEX for total bit-count below FrameBitRateMin - First 1/8 Region. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin; i.e., In the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3), \text{FrameBitRateMin}]$.</p>
	23:16	<p>Frame Delta QIndex Min[2] This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin - Below 1/8 and Above 1/4. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin; i.e., In the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 3)]$.</p>
	15:8	<p>Frame Delta QIndex Min[1] This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin - Below 1/4 and Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin; i.e., In the range of $[(\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 1), (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 2)]$.</p>
	7:0	<p>Frame Delta QIndex Min[0] This field is the Frame Level Delta QINDEX for bit-count below FrameBitRateMin - Below 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMin; i.e., In the range of $[0, (\text{FrameBitRateMin} - \text{FrameBitRateMinDelta} \gg 1)]$.</p>

MFX_VP8_Encoder_CFG				
7	31:0	Per Segment Frame Delta QIndex Max[1]		
8	31:0	Per Segment Frame Delta QIndex Min[1]		
9	31:0	Per Segment Frame Delta QIndex Max[2]		
10	31:0	Per Segment Frame Delta QIndex Min[2]		
11	31:0	Per Segment Frame Delta QIndex Max[3]		
12	31:0	Per Segment Frame Delta QIndex Min[3]		
13	31:24	<p>Frame Delta Loop Filter Max[3]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame level delta LoopFilter for total bit-count above FrameBitRateMax - First 1/8 region. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax.i.e., in the range of (FrameBitRateMax, (FrameBitRateMax+ FrameBitRateMaxDelta » 3)].</p>	Format:	U8
Format:	U8			
	23:16	<p>Frame Delta Loop Filter Max[2]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame level delta LoopFilter for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above FrameBitRateMax.i.e., in the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3) and (FrameBitRateMax + FrameBitRateMaxDelta » 2)].</p>	Format:	U8
Format:	U8			
	15:8	<p>Frame Delta Loop Filter Max[1]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above 1/4 and Below 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/4 and 1/2 of FrameBitRateMaxDelta above FrameBitRateMax.i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta » 2) and (FrameBitRateMax+ FrameBitRateMaxDelta » 1)].</p>	Format:	U8
Format:	U8			
	7:0	<p>Frame Delta Loop Filter Max[0]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of</p>	Format:	U8
Format:	U8			

MFX_VP8_Encoder_CFG				
		FrameBitRateMaxDelta.i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta » 1), infinite).		
14	31:24	<p>Frame Delta Loop Filter Min[3]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame level delta LOOPFILTER for total bit-count below FrameBitRateMin - First 1/8 region. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin.i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 3), FrameBitRateMin).</p>	Format:	U8
	Format:	U8		
	23:16	<p>Frame Delta Loop Filter Min[2]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame level delta LOOPFILTER for bit-count below FrameBitRateMin - Below 1/ 8 and Above 1/4. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin.i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 2), (FrameBitRateMin - FrameBitRateMinDelta » 3)].</p>	Format:	U8
	Format:	U8		
15:8	<p>Frame Delta Loop Filter Min[1]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame level delta LOOPFILTER for bit-count below FrameBitRateMin- Below 1/4 and Above 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin.i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 1) and (FrameBitRateMin - FrameBitRateMinDelta » 2)].</p>	Format:	U8	
Format:	U8			
7:0	<p>Frame Delta Loop Filter Min[0]</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U8</td> </tr> </table> <p>This field is the Frame Level Delta LOOPFILTER for bit-count below FrameBitRateMin - Below 1/ 2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta.i.e., in the range of [0, (FrameBitRateMin - FrameBitRateMinDelta » 1).</p>	Format:	U8	
Format:	U8			
15	31:0	Per Segment Frame Delta LoopFilter Max[1]		

MFX_VP8_Encoder_CFG								
16	31:0	Per Segment Frame Delta LoopFilter Min[1]						
17	31:0	Per Segment Frame Delta LoopFilter Max[2]						
18	31:0	Per Segment Frame Delta LoopFilter Min[2]						
19	31:0	Per Segment Frame Delta LoopFilter Max[3]						
20	31:0	Per Segment Frame Delta LoopFilter Min[3]						
21	31	Reserved Format: _____ MBZ						
	30:16	FrameBitRateMinDelta Format: _____ U15 This field is used to select the frame delta QINDEX when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0-4095]</td> <td></td> <td>When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.</td> </tr> </tbody> </table>	Value	Name	Description	[0-4095]		When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.
	Value	Name	Description					
	[0-4095]		When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.					
	15	Reserved Format: _____ MBZ						
14:0	Frame Bit Rate Max Delta Format: _____ U15 This field is used to select the frame delta QINDEX when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>[0-4095]</td> <td></td> <td>When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.</td> </tr> </tbody> </table>	Value	Name	Description	[0-4095]		When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.	
Value	Name	Description						
[0-4095]		When FrameBitrateMinUnit is in Bytes, this range is in Bytes. When FrameBitrateMinUnit is in KB, this range is in KB units.						
22	31:24	Reserved Format: _____ MBZ						
22	23	Show Frame Format: _____ U1 VP8 Frame Tag, Show Frame Field						
	22:20	Bitstream Format Version Format: _____ U3 VP8 Frame Tag, Version Field						
	19:18	Reserved Format: _____ MBZ						
	17:16	Min Frame WSize Unit Format: _____ U2						

MFX_VP8_Encoder_CFG											
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 35%;">Name</th> <th style="width: 50%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Compatibility Mode</td> <td>MinFrameWSizeUnit is in old mode (128b/16Kb)</td> </tr> <tr> <td>1h</td> <td>New Mode</td> <td>MinFrameWSizeUnit is in new mode (32byte/4Kb)</td> </tr> </tbody> </table>	Value	Name	Description	0h	Compatibility Mode	MinFrameWSizeUnit is in old mode (128b/16Kb)	1h	New Mode	MinFrameWSizeUnit is in new mode (32byte/4Kb)
Value	Name	Description									
0h	Compatibility Mode	MinFrameWSizeUnit is in old mode (128b/16Kb)									
1h	New Mode	MinFrameWSizeUnit is in new mode (32byte/4Kb)									
	15:0	<p>Min Frame WSize</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Encoder Only</td> </tr> </table> <p>This field (in Word, 16-bit) is specified to compensate for Intel Rate Control. Zero padding would be performed.</p>	Exists If:	//Encoder Only							
Exists If:	//Encoder Only										
23	31:16	<p>Vertical_Size_Code</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>U16</td> </tr> </table> <p>Frame Tag Vertical Size Code, composed of{VerticalScale[15:14], FrameHeight[13:0]}</p>	Format:	U16							
Format:	U16										
	15:0	<p>Horizontal_Size_Code</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>U16</td> </tr> </table> <p>Frame Tag Horizontal Size Code, composed of{HorizontalScale[15:14], FrameWidth[13:0]}</p>	Format:	U16							
Format:	U16										
24	31:0	<p>Frame Header Bit Count</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>U32</td> </tr> </table> <p>Binarized Header Bit Count.</p>	Format:	U32							
Format:	U32										
25	31:0	<p>Frame Header Bin Buffer Qindex Update Pointer</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>U32</td> </tr> </table> <p>Binarized Header Qindex Update PointerIf Segment Enabled and UpdateSegmentFeature enabled, 4 per segment Qindices would be updated in Binarized header (Only ABS mode supported).Else Base Qindex would be updated</p>	Format:	U32							
Format:	U32										
26	31:0	<p>Frame Header Bin Buffer LoopFilter Update Pointer</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>U32</td> </tr> </table> <p>Binarized Header LoopFilter Update PointerIf Segment Enabled and UpdateSegmentFeature enabled, 4 per segment LoopFilters would be updated in Binarized header (Only ABS mode supported).ElseBase LoopFilter would be updated.</p>	Format:	U32							
Format:	U32										
27	31:0	<p>Frame Header Bin Buffer Token Update Pointer</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>U32</td> </tr> </table> <p>Binarized Header TokenUpdate Pointer</p>	Format:	U32							
Format:	U32										
28	31:0	<p>Frame Header Bin Buffer MVUpdate Pointer</p>									

MFX_VP8_Encoder_CFG																																				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U32</td> </tr> </table> <p>Binarized Header MVUpdate Pointer.</p>	Format:	U32																																
Format:	U32																																			
<p style="text-align: center;">29</p> <p>Programming</p> <p>Notes: The only value permitted for CV7 through CV0 is 0xf</p>	31:28	ClampValues - CV7																																		
	27:24	CV6																																		
	23:20	CV5																																		
	19:16	CV4																																		
	15:12	CV3																																		
	11:8	CV2																																		
	7:4	CV1																																		
	3:0	CV0 - Clamp Value 0																																		
		<table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U4</td> </tr> </table> <p>If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with 2CV0-1. For coefficients at locations marked as 'none', no clamping is performed. The following mappings are only applied to luma and chroma blocks\subblocks containing AC coefficients (blocks\subblocks with only DC coeffs will not be clamped).</p> <p>For 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="width: 100%; text-align: center;"> <tr><td>none</td><td>CV7</td><td>CV5</td><td>CV4</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV4</td><td>CV3</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV1</td></tr> <tr><td>CV4</td><td>CV3</td><td>CV1</td><td>CV0</td></tr> </table> <p>For 4x4 field block, each coefficient is mapped to one of the eight CV values as following:</p> <table border="1" style="width: 100%; text-align: center;"> <tr><td>none</td><td>CV6</td><td>CV3</td><td>CV1</td></tr> <tr><td>CV7</td><td>CV6</td><td>CV3</td><td>CV1</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV0</td></tr> <tr><td>CV5</td><td>CV4</td><td>CV2</td><td>CV0</td></tr> </table>	Format:	U4	none	CV7	CV5	CV4	CV7	CV6	CV4	CV3	CV5	CV4	CV2	CV1	CV4	CV3	CV1	CV0	none	CV6	CV3	CV1	CV7	CV6	CV3	CV1	CV5	CV4	CV2	CV0	CV5	CV4	CV2	CV0
	Format:	U4																																		
none	CV7	CV5	CV4																																	
CV7	CV6	CV4	CV3																																	
CV5	CV4	CV2	CV1																																	
CV4	CV3	CV1	CV0																																	
none	CV6	CV3	CV1																																	
CV7	CV6	CV3	CV1																																	
CV5	CV4	CV2	CV0																																	
CV5	CV4	CV2	CV0																																	
		<table border="1" style="width: 100%;"> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> <tr> <td style="text-align: center;">0-15</td> <td></td> </tr> </table>	Value	Name	0-15																															
Value	Name																																			
0-15																																				

MFX_VP8_PAK_OBJECT

MFX_VP8_PAK_OBJECT			
Source:	VideoCS		
Length Bias:	2		
<p>The MFX_VP8_PAK_OBJECT command is the second primitive command for the VP8 Encoding Pipeline. The MV Data portion of the bitstream is loaded as indirect data object. Before issuing a MFX_VP8_PAK_OBJECT command, all VP8 MFX states need to be valid; therefore the commands used to set these states need to have been issued prior to the issue of this command. MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the first MB. MFX_VP8_PAK_OBJECT command follows the MbType definition like MFD. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_VP8_PAK_OBJECT
		Format:	OpCode
	26:24	Media Command Opcode	
		Default Value:	4h VP8_ENC
		Format:	OpCode
	23:21	SubOpcode A	
Default Value:		2h	
Format:		OpCode	
20:16	SubOpcode B		
	Default Value:	9h	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Default Value:	5h DWORD_COUNT_n	
	Format:	=n	
1	31:30	Reserved	
		Format:	MBZ
	29	Enable Inline MV data	
		Format:	Enable

MFV_VP8_PAK_OBJECT						
		This field denotes if the MV data will be sent inline following the other inline data instead of being indirect.				
	28:10	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
	9:0	Indirect PAK-MV Data Length <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U10</td> </tr> </table> <p>This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size). Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.</p>	Format:	U10		
Format:	U10					
2	31:29	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
	28:0	Indirect PAK-MV Data Start Address Offset <p>This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address. Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 60%; text-align: center;">Value</th> <th style="width: 40%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0,512MB)</td> <td></td> </tr> </tbody> </table>	Value	Name	[0,512MB)	
Value	Name					
[0,512MB)						
3..6	127:0	Inline Data <p>All the required MB level controls and parameters for encoding are captured as Inline Data Description - VP8 PAK OBJECT. It has a fixed size of 4 DWs. Its definition is described in the next section.</p>				

MFX_VP8_PIC_STATE

MFX_VP8_PIC_STATE											
Source:	VideoCS										
Length Bias:	2										
This must be the very first command to issue after the surface state, the pipe select and base address setting commands and must be issued before MFX_VP8_IMG_STATE.											
DWord	Bit	Description									
0	31:29	Command Type									
		Default Value:	3h PARALLEL_VIDEO_PIPE								
		Format:	OpCode								
	28:27	Pipeline									
		Default Value:	2h Video Codec								
		Format:	OpCode								
	26:24	Media Command OpCode									
		Default Value:	4h VP8								
		Format:	OpCode								
	23:21	Sub OpCode A									
Default Value:		0h VP8 Common									
Format:		OpCode									
20:16	Sub OpCode B										
	Default Value:	0h MFX_VP8_PIC_STATE									
	Format:	OpCode									
15:12	Reserved										
	Format:	MBZ									
11:0	DWord Length	Format:	=n								
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>000h</td> <td>Excludes DWord (0,1) [Default]</td> <td>A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware."</td> </tr> <tr> <td>024h</td> <td></td> <td>Used for normal decode and encode mode</td> </tr> </tbody> </table>		Value	Name	Description	000h	Excludes DWord (0,1) [Default]	A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware."	024h	
	Value	Name	Description								
	000h	Excludes DWord (0,1) [Default]	A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware."								
024h		Used for normal decode and encode mode									
1	31:24	Reserved									
		Format:	MBZ								
1	23:16	Frame Height Minus 1									
		Exists If:	//Decoder / Encoder								

MFX_VP8_PIC_STATE											
	<table border="1"> <tr> <td>Format:</td> <td>U8</td> </tr> <tr> <td colspan="2">Picture Height in integer number of MBs minus 1, so the min pic height can be program is 16 rows of pixels.</td> </tr> </table>	Format:	U8	Picture Height in integer number of MBs minus 1, so the min pic height can be program is 16 rows of pixels.							
	Format:	U8									
	Picture Height in integer number of MBs minus 1, so the min pic height can be program is 16 rows of pixels.										
	15:8 Reserved										
<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ										
7:0 Frame Width Minus 1											
	<table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> <tr> <td colspan="2">Picture Width in integer number of MBs minus 1, so the min pic width can be program is 16 pixels.</td> </tr> </table>	Exists If:	//Decoder / Encoder	Format:	U8	Picture Width in integer number of MBs minus 1, so the min pic width can be program is 16 pixels.					
Exists If:	//Decoder / Encoder										
Format:	U8										
Picture Width in integer number of MBs minus 1, so the min pic width can be program is 16 pixels.											
2	31:26 Reserved										
	<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
	Format:	MBZ									
	25:24 Log2 Num of Partition										
	<table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U2</td> </tr> </table>	Exists If:	//Decoder / Encoder	Format:	U2						
	Exists If:	//Decoder / Encoder									
	Format:	U2									
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1 Token partition</td> </tr> <tr> <td>1</td> <td>2 Token partition</td> </tr> <tr> <td>2</td> <td>4 Token partition</td> </tr> <tr> <td>3</td> <td>8 Token partition</td> </tr> </tbody> </table>	Value	Name	0	1 Token partition	1	2 Token partition	2	4 Token partition	3	8 Token partition
	Value	Name									
	0	1 Token partition									
1	2 Token partition										
2	4 Token partition										
3	8 Token partition										
23:19 Reserved											
<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ										
18:16 Deblock Sharpness Level											
<table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U3</td> </tr> <tr> <td colspan="2">Specify the sharpness level, as one of the regular deblocking strength control parameters.</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Set to 0 to disable the use of sharpness control.</td> </tr> </table>	Exists If:	//Decoder / Encoder	Format:	U3	Specify the sharpness level, as one of the regular deblocking strength control parameters.		Programming Notes		Set to 0 to disable the use of sharpness control.		
Exists If:	//Decoder / Encoder										
Format:	U3										
Specify the sharpness level, as one of the regular deblocking strength control parameters.											
Programming Notes											
Set to 0 to disable the use of sharpness control.											
15:14 Reserved											
<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ										
13 Alternate Ref Pic MV SignBias Flag											
<table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td colspan="2">Alternate Reference Picture MV sign bias flag, specified for non-key frame only.</td> </tr> </table>	Exists If:	//Decoder / Encoder	Alternate Reference Picture MV sign bias flag, specified for non-key frame only.								
Exists If:	//Decoder / Encoder										
Alternate Reference Picture MV sign bias flag, specified for non-key frame only.											
12 Golden Ref Picture MV SignBias Flag											

MFX_VP8_PIC_STATE

	Exists If:	//Decoder / Encoder	Golden Reference Picture MV sign bias flag, specified for non-key frame only.
11	Mode Reference Loop Filter Delta Enabled		
	Exists If:	//Decoder / Encoder	
	Value	Name	Description
	0		Mode or Reference Loop Filter Delta Adjustment for current frame is disabled.
	1		Mode or Reference Loop Filter Delta Adjustment for current frame is enabled.
10	MB NoCoeff SkipFlag		
	Exists If:	//Decoder / Encoder	Frame level control if Skip MB (with no non-zero coefficient) is allowed or not.
	Value	Name	Description
	0		All MBs will have its MB level signaling mb_skip_coeff forced to 0. That is, no skip of coefficient record in the bitstream (even their values are all 0s)
	1		Skip MB is enabled in the per MB record.
9	Update MBSegment Map Flag		
	Exists If:	//Decoder / Encoder	
	Value	Name	Description
	0		Disable segmentation update
	1		Enable segmentation update, and to enable reading segment_id for each MB.
8	Segment Enable Flag		
	Exists If:	//Decoder / Encoder	
	Value	Name	Description
	0		Disable Segmentation processing in the current frame
	1		Enable Segmentation processing in the current frame
7	Segmentation ID StreamIn Enable		
	Exists If:	//Decoder Only	
	Value	Name	
	0		StreamIn Disabled
	1		StreamIn Enabled
	Programming Notes		
	When 0, no input needed.		
6	Segmentation ID StreamOut Enable		

MFX_VP8_PIC_STATE																				
		<table border="1"> <tr> <td>Exists If:</td> <td colspan="2">//Decoder Only</td> </tr> <tr> <td></td> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> </tr> <tr> <td></td> <td>0</td> <td>StreamOut Disabled</td> </tr> <tr> <td></td> <td>1</td> <td>StreamOut Enabled</td> </tr> <tr> <td colspan="3" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="3">When 0, no output needed.</td> </tr> </table>	Exists If:	//Decoder Only			Value	Name		0	StreamOut Disabled		1	StreamOut Enabled	Programming Notes			When 0, no output needed.		
Exists If:	//Decoder Only																			
	Value	Name																		
	0	StreamOut Disabled																		
	1	StreamOut Enabled																		
Programming Notes																				
When 0, no output needed.																				
	5	<table border="1"> <tr> <td colspan="3">sKeyFrameFlag</td> </tr> <tr> <td>Exists If:</td> <td colspan="2">//Decoder / Encoder</td> </tr> <tr> <td></td> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> </tr> <tr> <td></td> <td>0</td> <td>Non-Key Frame (P-Frame)</td> </tr> <tr> <td></td> <td>1</td> <td>Key Frame (I-Frame)</td> </tr> </table>	sKeyFrameFlag			Exists If:	//Decoder / Encoder			Value	Name		0	Non-Key Frame (P-Frame)		1	Key Frame (I-Frame)			
sKeyFrameFlag																				
Exists If:	//Decoder / Encoder																			
	Value	Name																		
	0	Non-Key Frame (P-Frame)																		
	1	Key Frame (I-Frame)																		
	4	<table border="1"> <tr> <td colspan="3">DBLKFilterType</td> </tr> <tr> <td>Exists If:</td> <td colspan="2">//Decoder / Encoder</td> </tr> <tr> <td colspan="3">To specify VP8 Profile of operation.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> <td style="text-align: center;">Description</td> </tr> <tr> <td>0</td> <td></td> <td>Use a full feature normal deblocking filter</td> </tr> <tr> <td>1</td> <td></td> <td>Use a simple filter for deblocking</td> </tr> </table>	DBLKFilterType			Exists If:	//Decoder / Encoder		To specify VP8 Profile of operation.			Value	Name	Description	0		Use a full feature normal deblocking filter	1		Use a simple filter for deblocking
DBLKFilterType																				
Exists If:	//Decoder / Encoder																			
To specify VP8 Profile of operation.																				
Value	Name	Description																		
0		Use a full feature normal deblocking filter																		
1		Use a simple filter for deblocking																		
	3:2	<table border="1"> <tr> <td colspan="3">Reserved</td> </tr> <tr> <td>Format:</td> <td colspan="2">MBZ</td> </tr> </table>	Reserved			Format:	MBZ													
Reserved																				
Format:	MBZ																			
	1	<table border="1"> <tr> <td colspan="3">Chroma Full Pixel MC Filter Mode</td> </tr> <tr> <td>Exists If:</td> <td colspan="2">//Decoder / Encoder</td> </tr> <tr> <td colspan="3">To specify VP8 Profile of operation.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> <td style="text-align: center;">Description</td> </tr> <tr> <td>0</td> <td></td> <td>Chroma MC filter operates in sub-pixel mode</td> </tr> <tr> <td>1</td> <td></td> <td>Chroma MC filter only operates in full pixel position, i.e. no sub-pixel interpolation.</td> </tr> </table>	Chroma Full Pixel MC Filter Mode			Exists If:	//Decoder / Encoder		To specify VP8 Profile of operation.			Value	Name	Description	0		Chroma MC filter operates in sub-pixel mode	1		Chroma MC filter only operates in full pixel position, i.e. no sub-pixel interpolation.
Chroma Full Pixel MC Filter Mode																				
Exists If:	//Decoder / Encoder																			
To specify VP8 Profile of operation.																				
Value	Name	Description																		
0		Chroma MC filter operates in sub-pixel mode																		
1		Chroma MC filter only operates in full pixel position, i.e. no sub-pixel interpolation.																		
	0	<table border="1"> <tr> <td colspan="3">MC Filter Select</td> </tr> <tr> <td>Exists If:</td> <td colspan="2">//Decoder / Encoder</td> </tr> <tr> <td colspan="3">To specify VP8 Profile of operation.</td> </tr> <tr> <td style="text-align: center;">Value</td> <td style="text-align: center;">Name</td> <td style="text-align: center;">Description</td> </tr> <tr> <td>0</td> <td></td> <td>6-tap filter (regular filter mode)</td> </tr> <tr> <td>1</td> <td></td> <td>2-tap bilinear filter (simple profile/version mode)</td> </tr> </table>	MC Filter Select			Exists If:	//Decoder / Encoder		To specify VP8 Profile of operation.			Value	Name	Description	0		6-tap filter (regular filter mode)	1		2-tap bilinear filter (simple profile/version mode)
MC Filter Select																				
Exists If:	//Decoder / Encoder																			
To specify VP8 Profile of operation.																				
Value	Name	Description																		
0		6-tap filter (regular filter mode)																		
1		2-tap bilinear filter (simple profile/version mode)																		
3	31:30	<table border="1"> <tr> <td colspan="3">Reserved</td> </tr> <tr> <td>Format:</td> <td colspan="2">MBZ</td> </tr> </table>	Reserved			Format:	MBZ													
Reserved																				
Format:	MBZ																			

MFX_VP8_PIC_STATE

	29:24	DBLKFilterLevel for Segment3	
		Exists If:	//Decoder / Encoder
		Format:	U6
		Value	Name
		Description	
		0	Signifies disable in loop deblocking operation
			This is used to set a VP8 profile without in loop deblocker.
		Programming Notes	
		There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.	
	23:22	Reserved	
		Format:	MBZ
	21:16	DBLKFilterLevel for Segment2	
		Exists If:	//Decoder / Encoder
		Format:	U6
		Value	Name
		Description	
		0	Signifies disable in loop deblocking operation
			This is used to set a VP8 profile without in loop deblocker.
		Programming Notes	
		There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.	
	15:14	Reserved	
		Format:	MBZ
	13:8	DBLKFilterLevel for Segment1	
		Exists If:	//Decoder / Encoder
		Format:	U6
		Value	Name
		Description	
		0	Signifies disable in loop deblocking operation
			This is used to set a VP8 profile without in loop deblocker.
		Programming Notes	
		There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.	
	7:6	Reserved	
		Format:	MBZ

MFX_VP8_PIC_STATE												
4	5:0	<p>DBLKFilterLevel for Segment0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 45%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Signifies disable in loop deblocking operation</td> <td>This is used to set a VP8 profile without in loop deblocker.</td> </tr> </tbody> </table> <div style="border: 1px solid black; background-color: #e6f2ff; padding: 5px; margin-top: 5px; text-align: center;"> <p>Programming Notes</p> </div> <p>There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.</p>	Exists If:	//Decoder / Encoder	Format:	U6	Value	Name	Description	0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.
	Exists If:	//Decoder / Encoder										
	Format:	U6										
	Value	Name	Description									
	0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.									
	31	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
	Format:	MBZ										
	30:24	<p>Seg 3 Qindex</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U7</td> </tr> </table> <p>Quantizer Value for Segment ID 3</p>	Exists If:	//Encoder Only	Format:	U7						
	Exists If:	//Encoder Only										
	Format:	U7										
23	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
22:16	<p>Seg 2 Qindex</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U7</td> </tr> </table> <p>Quantizer Value for Segment ID 2</p>	Exists If:	//Encoder Only	Format:	U7							
Exists If:	//Encoder Only											
Format:	U7											
15	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
14:8	<p>Seg 1 Qindex</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U7</td> </tr> </table> <p>Quantizer Value for Segment ID 1</p>	Exists If:	//Encoder Only	Format:	U7							
Exists If:	//Encoder Only											
Format:	U7											
7	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ											
6:0	<p>Seg 0 Qindex</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U7</td> </tr> </table> <p>Quantizer Value for Segment ID 0.</p> <div style="border: 1px solid black; background-color: #e6f2ff; padding: 5px; margin-top: 5px; text-align: center;"> <p>Programming Notes</p> </div>	Exists If:	//Encoder Only	Format:	U7							
Exists If:	//Encoder Only											
Format:	U7											

MFV_VP8_PIC_STATE		
		This is the [Default] Qindex
5	31:29	Reserved Format: MBZ
	28	UVac Qindex Delta Sign Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVac
	27:24	UVac QindexDelta Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVac
	23:21	Reserved Format: MBZ
	20	UVdc Qindex Delta Sign Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVdc
	19:16	UVdc Qindex Delta Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVdc
	15:13	Reserved Format: MBZ
	12	Y2ac Qindex Sign Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2ac
	11:8	Y2ac Qindex Delta Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2ac
	7:5	Reserved

MFX_VP8_PIC_STATE		
		Format: MBZ
6	4	Y2ac Qindex Delta Sign Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2dc This is the [Default] Qindex Delta Sign
	3:0	Y2dc Qindex Delta Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2dc This is the [Default] Qindex Delta
	31:5	Reserved Format: MBZ
	4	Y1dc Qindex Delta Sign Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y1dc This is the [Default] Qindex Delta Sign
	3:0	Y1dc Qindex Delta Exists If: //Encoder Only Absolute Quantization index delta for Y1dc This is the [Default] Qindex Delta
	31:15	Reserved Format: MBZ
	14:8	Clamp Qindex high Exists If: //Encoder Only Format: U7 Maximum Clamp Value for Qindex used in quantization.
	7	Reserved Format: MBZ
	6:0	Clamp Qindex Low Exists If: //Encoder Only

MFX_VP8_PIC_STATE								
		<table border="1"> <tr> <td>Format:</td> <td>U7</td> </tr> <tr> <td colspan="2">Minimum Clamp Value for Qindex used in quantization.</td> </tr> </table>	Format:	U7	Minimum Clamp Value for Qindex used in quantization.			
Format:	U7							
Minimum Clamp Value for Qindex used in quantization.								
8	31:25	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	24:16	Quantizer Value [1][BlockType3=UVAC] <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>U9</td> </tr> <tr> <td colspan="2">Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]</td> </tr> </table>	Exists If:	//Decoder Only	Format:	U9	Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	
		Exists If:	//Decoder Only					
		Format:	U9					
Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]								
15:9	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ					
Format:	MBZ							
8:0	Quantizer Value [1][BlockType2=UVDC] <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td colspan="2">Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]</td> </tr> </table>	Exists If:	//Decoder Only	Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]				
Exists If:	//Decoder Only							
Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]								
9	31:25	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	24:16	Quantizer Value [1][BlockType5=Y2AC] <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>U9</td> </tr> <tr> <td colspan="2">Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]</td> </tr> </table>	Exists If:	//Decoder Only	Format:	U9	Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	
		Exists If:	//Decoder Only					
		Format:	U9					
Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]								
15:9	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ					
Format:	MBZ							
8:0	Quantizer Value [1][BlockType4=Y2DC] <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td colspan="2">Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]</td> </tr> </table>	Exists If:	//Decoder Only	Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]				
Exists If:	//Decoder Only							
Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]								
10	31:25	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
	Format:	MBZ						
	24:16	Quantizer Value [2][BlockType1=Y1AC] <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>U9</td> </tr> <tr> <td colspan="2">Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]</td> </tr> </table>	Exists If:	//Decoder Only	Format:	U9	Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	
		Exists If:	//Decoder Only					
Format:		U9						
Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]								
15:9	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ					
Format:	MBZ							

MFX_VP8_PIC_STATE

	8:0	Quantizer Value [2][BlockType0=Y1DC] Exists If: //Decoder Only Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
11	31:25	Reserved Format: MBZ
	24:16	Quantizer Value [2][BlockType3=UVAC] Exists If: //Decoder Only Format: U9 Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
	15:9	Reserved Format: MBZ
	8:0	Quantizer Value [2][BlockType2=UVDC] Exists If: //Decoder Only Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
12	31:25	Reserved Format: MBZ
	24:16	Quantizer Value [2][BlockType5=Y2AC] Exists If: //Decoder Only Format: U9 Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
	15:9	Reserved Format: MBZ
	8:0	Quantizer Value [2][BlockType4=Y2DC] Exists If: //Decoder Only Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
13	31:25	Reserved Format: MBZ
	24:16	Quantizer Value [3][BlockType1=Y1AC] Exists If: //Decoder Only Format: U9 Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
	15:9	Reserved Format: MBZ

MFX_VP8_PIC_STATE						
	8:0	Quantizer Value [3][BlockType0=Y1DC] <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Decoder Only</td> </tr> </table> Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	Exists If:	//Decoder Only		
Exists If:	//Decoder Only					
14	31:25	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ				
	24:16	Quantizer Value [3][BlockType3=UVAC] <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>U9</td> </tr> </table> Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	Exists If:	//Decoder Only	Format:	U9
	Exists If:	//Decoder Only				
Format:	U9					
15:9	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
8:0	Quantizer Value [3][BlockType2=UVDC] <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Decoder Only</td> </tr> </table> Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	Exists If:	//Decoder Only			
Exists If:	//Decoder Only					
15	31:25	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ				
	24:16	Quantizer Value [3][BlockType5=Y2AC] <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>U9</td> </tr> </table> Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	Exists If:	//Decoder Only	Format:	U9
	Exists If:	//Decoder Only				
Format:	U9					
15:9	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
8:0	Quantizer Value [3][BlockType4=Y2DC] <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Decoder Only</td> </tr> </table> Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]	Exists If:	//Decoder Only			
Exists If:	//Decoder Only					
16	31:6	CoeffProbability StreamIn Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Exists If:</td> <td>//Decoder Only</td> </tr> <tr> <td>Format:</td> <td>SplitBaseAddress4KByteAligned</td> </tr> </table> <p>It is specified for non-key frame only. It is the final computed probability table for parsing Coeff in the bitstream. The buffer is unsigned 8-bit * 1056 entries (CoeffProbs[4][8][3][11].</p>	Exists If:	//Decoder Only	Format:	SplitBaseAddress4KByteAligned
	Exists If:	//Decoder Only				
Format:	SplitBaseAddress4KByteAligned					
5:0	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
17	31:16	Reserved				

MFX_VP8_PIC_STATE

		Format:	MBZ
	15:0	CoeffProbability StreamIn Address	
		Exists If:	//Decoder Only
		This field is for the upper range of CoeffProbability StreamIn Address	
18	31:15	Reserved	
		Format:	MBZ
	14:13	CoeffProbability StreamIn - Tiled Resource Mode	
		Exists If:	//Decoder Only
		Format:	U2
		For Media Surfaces: This field specifies the tiled resource mode.	
		Value	Name
		0h	TRMODE_NONE
		1h	TRMODE_TILEYF
		2h	TRMODE_TILEYS
		3h	Reserved
	12:11	Reserved	
		Format:	MBZ
	10	CoeffProbability StreamIn - Memory Compression Mode	
		Exists If:	//Decoder Only
		Format:	U1
		Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter, Media Memory Compression section for more details.	
		Value	Name
		0	Horizontal Compression Mode
		1	Vertical Compression Mode
	9	CoeffProbability StreamIn - Memory Compression Enable	
		Exists If:	//Decoder Only
		Format:	Enable
		Memory compression will be attempted for this surface.	
	8:7	CoeffProbability StreamIn - Arbitration Priority Control	
		Exists If:	//Decoder Only
		Format:	U2
		This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.	
		Value	Name
		00b	Highest priority

MFX_VP8_PIC_STATE									
	<table border="1"> <tr> <td>01b</td> <td>Second highest priority</td> </tr> <tr> <td>10b</td> <td>Third highest priority</td> </tr> <tr> <td>11b</td> <td>Lowest priority</td> </tr> </table>	01b	Second highest priority	10b	Third highest priority	11b	Lowest priority		
01b	Second highest priority								
10b	Third highest priority								
11b	Lowest priority								
6:1	<p>CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Encoder Only</td> </tr> <tr> <td>Format:</td> <td>U6</td> </tr> </table> <p>The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.</p>	Exists If:	//Encoder Only	Format:	U6				
Exists If:	//Encoder Only								
Format:	U6								
0	Reserved								
19	<table border="1"> <tr> <td>31:24</td> <td>Reserved</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	31:24	Reserved	Format:	MBZ				
31:24	Reserved								
Format:	MBZ								
23:16	<p>MBSegmentIDTreeProbs[2]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.</p>	Exists If:	//Decoder / Encoder	Format:	U8				
Exists If:	//Decoder / Encoder								
Format:	U8								
15:8	<p>MBSegmentIDTreeProbs[1]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.</p>	Exists If:	//Decoder / Encoder	Format:	U8				
Exists If:	//Decoder / Encoder								
Format:	U8								
7:0	<p>MBSegmentIDTreeProbs[0]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.</p>	Exists If:	//Decoder / Encoder	Format:	U8				
Exists If:	//Decoder / Encoder								
Format:	U8								
20	<table border="1"> <tr> <td>31:24</td> <td>MBNoCoeffSkipFalseProb</td> </tr> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> <tr> <td colspan="2">8-bit probability value for CPBAC parsing of the MBNoCoeffSkip Flag in the bistream.</td> </tr> </table>	31:24	MBNoCoeffSkipFalseProb	Exists If:	//Decoder / Encoder	Format:	U8	8-bit probability value for CPBAC parsing of the MBNoCoeffSkip Flag in the bistream.	
31:24	MBNoCoeffSkipFalseProb								
Exists If:	//Decoder / Encoder								
Format:	U8								
8-bit probability value for CPBAC parsing of the MBNoCoeffSkip Flag in the bistream.									
23:16	<p>IntraMBProb</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>8-bit probability value for CPBAC parsing of the intra or inter MB type flag in the bitstream.</p>	Exists If:	//Decoder / Encoder	Format:	U8				
Exists If:	//Decoder / Encoder								
Format:	U8								

MFX_VP8_PIC_STATE

	15:8	InterPredFromLastRefProb	Exists If:	//Decoder / Encoder
			Format:	U8
8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.				
	7:0	InterPredFromGRefRefProb	Exists If:	//Decoder / Encoder
			Format:	U8
8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.				
21	31:24	YModeProb[3]	Exists If:	//Decoder / Encoder
			Format:	U8
	YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.			
	23:16	YModeProb[2]	Exists If:	//Decoder / Encoder
		Format:	U8	
YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.				
15:8	YModeProb[1]	Exists If:	//Decoder / Encoder	
		Format:	U8	
	YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.			
	7:0	YModeProb[0]	Exists If:	//Decoder / Encoder
		Format:	U8	
YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.				
22	31:24	Reserved	Format:	MBZ
	23:16	UVModeProb[2]	Exists If:	//Decoder / Encoder
			Format:	U8
UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.				
15:8	UVModeProb[1]			

MFX_VP8_PIC_STATE		
		Exists If: //Decoder / Encoder
		Format: U8
		UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
	7:0	UVModeProb[0]
		Exists If: //Decoder / Encoder
		Format: U8
		UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
23	31:24	MVUpdateProbs[0][3]
		Exists If: //Decoder / Encoder
		Format: U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
23:16	23:16	MVUpdateProbs[0][2]
		Exists If: //Decoder / Encoder
		Format: U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
15:8	15:8	MVUpdateProbs[0][1]
		Exists If: //Decoder / Encoder
		Format: U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
7:0	7:0	MVUpdateProbs[0][0]
		Exists If: //Decoder / Encoder
		Format: U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
24	31:24	MVUpdateProbs[0][7]
		Exists If: //Decoder / Encoder
		Format: U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	MVUpdateProbs[0][6]

MFX_VP8_PIC_STATE

		Exists If:	//Decoder / Encoder
		Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].	
	15:8	MVUpdateProbs[0][5]	
		Exists If:	//Decoder / Encoder
		Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].	
	7:0	MVUpdateProbs[0][4]	
		Exists If:	//Decoder / Encoder
		Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].	
25	31:24	MVUpdateProbs[0][11]	
		Exists If:	//Decoder / Encoder
		Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.	
	23:16	MVUpdateProbs[0][10]	
		Exists If:	//Decoder / Encoder
		Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.	
	15:8	MVUpdateProbs[0][9]	
		Exists If:	//Decoder / Encoder
		Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.	
	7:0	MVUpdateProbs[0][8]	
		Exists If:	//Decoder / Encoder
		Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.	
26	31:24	MVUpdateProbs[0][15]	
		Exists If:	//Decoder / Encoder
		Format:	U8

MFX_VP8_PIC_STATE

		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].				
	23:16	<p>MVUpdateProbs[0][14]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].</p>	Exists If:	//Decoder / Encoder	Format:	U8
Exists If:	//Decoder / Encoder					
Format:	U8					
	15:8	<p>MVUpdateProbs[0][13]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].</p>	Exists If:	//Decoder / Encoder	Format:	U8
Exists If:	//Decoder / Encoder					
Format:	U8					
	7:0	<p>MVUpdateProbs[0][12]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].</p>	Exists If:	//Decoder / Encoder	Format:	U8
Exists If:	//Decoder / Encoder					
Format:	U8					
27	31:24	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
	23:16	<p>MVUpdateProbs[0][18]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].</p>	Exists If:	//Decoder / Encoder	Format:	U8
Exists If:	//Decoder / Encoder					
Format:	U8					
	15:8	<p>MVUpdateProbs[0][17]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].</p>	Exists If:	//Decoder / Encoder	Format:	U8
Exists If:	//Decoder / Encoder					
Format:	U8					
	7:0	<p>MVUpdateProbs[0][16]</p> <table border="1"> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>U8</td> </tr> </table> <p>MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].</p>	Exists If:	//Decoder / Encoder	Format:	U8
Exists If:	//Decoder / Encoder					
Format:	U8					

MFX_VP8_PIC_STATE

28	31:24	MVUpdateProbs[1][3]	Exists If:	//Decoder Only
			Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
29	23:16	MVUpdateProbs[1][2]	Exists If:	//Decoder Only
			Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
28	15:8	MVUpdateProbs[1][1]	Exists If:	//Decoder Only
			Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
28	7:0	MVUpdateProbs[1][0]	Exists If:	//Decoder Only
			Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
29	31:24	MVUpdateProbs[1][7]	Exists If:	//Decoder / Encoder
			Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	23:16	MVUpdateProbs[1][6]	Exists If:	//Decoder / Encoder
			Format:	U8
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	15:8	MVUpdateProbs[1][5]	Exists If:	//Decoder / Encoder
		Format:	U8	
	MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].			

MFX_VP8_PIC_STATE

		MFX_VP8_PIC_STATE	
	7:0	MVUpdateProbs[1][4]	
		Exists If:	//Decoder / Encoder
		Format:	U8
	MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
30	31:24	MVUpdateProbs[1][11]	
		Exists If:	//Decoder / Encoder
		Format:	U8
	MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	23:16	MVUpdateProbs[1][10]	
		Exists If:	//Decoder / Encoder
		Format:	U8
	MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	15:8	MVUpdateProbs[1][9]	
		Exists If:	//Decoder / Encoder
		Format:	U8
	MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
7:0	MVUpdateProbs[1][8]		
	Exists If:	//Decoder / Encoder	
	Format:	U8	
MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].			
31	31:24	MVUpdateProbs[1][15]	
		Exists If:	//Decoder / Encoder
		Format:	U8
	MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	23:16	MVUpdateProbs[1][14]	
		Exists If:	//Decoder / Encoder
	Format:	U8	

MFX_VP8_PIC_STATE

		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	15:8	MVUpdateProbs[1][13]		
		Exists If:	//Decoder / Encoder	
		Format:	U8	
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	7:0	MVUpdateProbs[1][12]		
		Exists If:	//Decoder / Encoder	
		Format:	U8	
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
32	31:24	Reserved		
		Format:	MBZ	
	23:16	MVUpdateProbs[1][18]		
		Exists If:	//Decoder / Encoder	
		Format:	U8	
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
	15:8	MVUpdateProbs[1][17]		
		Exists If:	//Decoder / Encoder	
		Format:	U8	
		MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].		
		7:0	MVUpdateProbs[1][16]	
			Exists If:	//Decoder / Encoder
		Format:	U8	
	MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].			
33	31	Reserved		
		Format:	MBZ	
	30:24	RefLFDelta3 (for ALTREF FRAME)		
		Exists If:	//Decoder / Encoder	
		Format:	S6	

MFX_VP8_PIC_STATE

		<p>Delta value for reference frame based adjustment of the MB-level's filter level value.</p> <p>RefLFDeltas [ref_frametype = 0 to 3]</p>				
		Programming Notes				
		<p>Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</p>				
23	Reserved	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
22:16	RefLFDelta2 (for GOLDEN FRAME)	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>S6</td> </tr> </table> <p>Delta value for reference frame based adjustment of the MB-level's filter level value.</p> <p>RefLFDeltas [ref_frametype = 0 to 3]</p>	Exists If:	//Decoder / Encoder	Format:	S6
Exists If:	//Decoder / Encoder					
Format:	S6					
		Programming Notes				
		<p>Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</p>				
15	Reserved	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
14:8	RefLFDelta1 (for LAST FRAME)	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>S6</td> </tr> </table> <p>Delta value for reference frame based adjustment of the MB-level's filter level value.</p> <p>RefLFDeltas [ref_frametype = 0 to 3]</p>	Exists If:	//Decoder / Encoder	Format:	S6
Exists If:	//Decoder / Encoder					
Format:	S6					
		Programming Notes				
		<p>Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</p>				
7	Reserved	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
6:0	RefLFDelta0 (for INTRA FRAME)	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>S6</td> </tr> </table> <p>Delta value for reference frame based adjustment of the MB-level's filter level value.</p>	Exists If:	//Decoder / Encoder	Format:	S6
Exists If:	//Decoder / Encoder					
Format:	S6					

MFX_VP8_PIC_STATE																
	<table border="1" style="width: 100%;"> <tr> <td colspan="2">RefLFDeltas [ref_frametype = 0 to 3]</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</td> </tr> </table>	RefLFDeltas [ref_frametype = 0 to 3]		Programming Notes		Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.										
RefLFDeltas [ref_frametype = 0 to 3]																
Programming Notes																
Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.																
34	31	<table border="1" style="width: 100%;"> <tr> <td colspan="2">Reserved</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Reserved		Format:	MBZ										
	Reserved															
	Format:	MBZ														
	30:24	<table border="1" style="width: 100%;"> <tr> <td colspan="2">ModelFDelta3 (for SPLITMV mode)</td> </tr> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>S6</td> </tr> <tr> <td colspan="2">Delta value for mode based adjustment of the MB-level's filter level value.</td> </tr> <tr> <td colspan="2">ModelFDeltas[MB_Type = 0 to 3]</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Please note that although ModelFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</td> </tr> </table>	ModelFDelta3 (for SPLITMV mode)		Exists If:	//Decoder / Encoder	Format:	S6	Delta value for mode based adjustment of the MB-level's filter level value.		ModelFDeltas[MB_Type = 0 to 3]		Programming Notes		Please note that although ModelFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.	
	ModelFDelta3 (for SPLITMV mode)															
	Exists If:	//Decoder / Encoder														
	Format:	S6														
Delta value for mode based adjustment of the MB-level's filter level value.																
ModelFDeltas[MB_Type = 0 to 3]																
Programming Notes																
Please note that although ModelFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.																
23	<table border="1" style="width: 100%;"> <tr> <td colspan="2">Reserved</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Reserved		Format:	MBZ											
Reserved																
Format:	MBZ															
22:16	<table border="1" style="width: 100%;"> <tr> <td colspan="2">ModelFDelta2 (for Nearest, Near and New mode)</td> </tr> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>S6</td> </tr> <tr> <td colspan="2">Delta value for mode based adjustment of the MB-level's filter level value.</td> </tr> <tr> <td colspan="2">ModelFDeltas[MB_Type = 0 to 3]</td> </tr> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> <tr> <td colspan="2">Please note that although ModelFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</td> </tr> </table>	ModelFDelta2 (for Nearest, Near and New mode)		Exists If:	//Decoder / Encoder	Format:	S6	Delta value for mode based adjustment of the MB-level's filter level value.		ModelFDeltas[MB_Type = 0 to 3]		Programming Notes		Please note that although ModelFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.		
ModelFDelta2 (for Nearest, Near and New mode)																
Exists If:	//Decoder / Encoder															
Format:	S6															
Delta value for mode based adjustment of the MB-level's filter level value.																
ModelFDeltas[MB_Type = 0 to 3]																
Programming Notes																
Please note that although ModelFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.																
15	<table border="1" style="width: 100%;"> <tr> <td colspan="2">Reserved</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Reserved		Format:	MBZ											
Reserved																
Format:	MBZ															
14:8	<table border="1" style="width: 100%;"> <tr> <td colspan="2">ModelFDelta1 (for ZEROMV mode)</td> </tr> <tr> <td>Exists If:</td> <td>//Decoder / Encoder</td> </tr> <tr> <td>Format:</td> <td>S6</td> </tr> <tr> <td colspan="2">Delta value for mode based adjustment of the MB-level's filter level value.</td> </tr> <tr> <td colspan="2">ModelFDeltas[MB_Type = 0 to 3]</td> </tr> </table>	ModelFDelta1 (for ZEROMV mode)		Exists If:	//Decoder / Encoder	Format:	S6	Delta value for mode based adjustment of the MB-level's filter level value.		ModelFDeltas[MB_Type = 0 to 3]						
ModelFDelta1 (for ZEROMV mode)																
Exists If:	//Decoder / Encoder															
Format:	S6															
Delta value for mode based adjustment of the MB-level's filter level value.																
ModelFDeltas[MB_Type = 0 to 3]																

MFX_VP8_PIC_STATE														
		<table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</td> </tr> </tbody> </table>	Programming Notes		Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.									
Programming Notes														
Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.														
	7	Reserved Format: MBZ												
	6:0	ModeLFDelta0 (for B_PRED mode) Exists If: //Decoder / Encoder Format: S6 Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] <table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.</td> </tr> </tbody> </table>	Programming Notes		Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.									
Programming Notes														
Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.														
35	31:0	Segmentation ID Stream Base Address Exists If: //Decoder Only Format: SplitBaseAddress4KByteAligned It is specified when SegmentationIDStreamInEnable or SegmentationIDStreamOutEnable is specified. <table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Each cache has only 8 bits for 4 segmentation ID from 4 continuous MBs.</td> </tr> </tbody> </table>	Programming Notes		Each cache has only 8 bits for 4 segmentation ID from 4 continuous MBs.									
Programming Notes														
Each cache has only 8 bits for 4 segmentation ID from 4 continuous MBs.														
36	31:16	Reserved Format: MBZ												
	15:0	Segmentation ID Stream Base Address [47:32] Exists If: //Decoder Only This field is for the upper range of Segmentation ID Stream Base Address												
37	31:15	Reserved Format: MBZ												
	14:13	Segmentation ID Stream - Tiled Resource Mode Exists If: //Decoder Only Format: U2 For Media Surfaces: This field specifies the tiled resource mode. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>TRMODE_NONE</td> <td>No tiled resource</td> </tr> <tr> <td>1h</td> <td>TRMODE_TILEYF</td> <td>4KB tiled resources</td> </tr> <tr> <td>2h</td> <td>TRMODE_TILEYS</td> <td>64KB tiled resources</td> </tr> </tbody> </table>	Value	Name	Description	0h	TRMODE_NONE	No tiled resource	1h	TRMODE_TILEYF	4KB tiled resources	2h	TRMODE_TILEYS	64KB tiled resources
Value	Name	Description												
0h	TRMODE_NONE	No tiled resource												
1h	TRMODE_TILEYF	4KB tiled resources												
2h	TRMODE_TILEYS	64KB tiled resources												

MFX_VP8_PIC_STATE

	3h	Reserved	
12:11	Reserved		
	Format:		MBZ
10	Segmentation ID Stream - Memory Compression Mode		
	Format:		U1
	Distinguishes Vertical from Horizontal compression. Please refer to vol1a Memory Data Formats chapter, Media Memory Compression section for more details.		
	Value	Name	
	0	Horizontal Compression Mode	
	1	Vertical Compression Mode	
9	Segmentation ID Stream - Memory Compression Enable		
	Format:		Enable
	Memory compression will be attempted for this surface.		
8:7	Segmentation ID Stream - Arbitration Priority Control		
	Exists If:	//Decoder Only	
	Format:		U2
	This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.		
	Value	Name	
	00b	Highest priority	
	01b	Second highest priority	
	10b	Third highest priority	
	11b	Lowest priority	
6:1	CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables		
	Format:		U6
	The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.		
0	Reserved		

MFX_WAIT

MFX_WAIT			
Source:	VideoCS		
Length Bias:	1		
<p>This command can be considered the same as an MI_NOOP except that the command parser will not parse the next command until the following happens</p> <ul style="list-style-type: none"> • AVC or VC1 BSD mode: The command will stall the parser until completion of the BSD object • IT, encoder, and MPEG2 BSD mode: The command will stall the parser until the object package is sent down the pipeline. This command should be used to ensure the preemption enable window occurs during the time the object command is being executed down the pipeline. 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	03h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Command Subtype	
		Default Value:	01h MFX_SINGLE_DW
		Format:	OpCode
	26:16	Sub-Opcode	
		Default Value:	0h MFX_WAIT
		Format:	OpCode
	15:10	Reserved	
		Format:	MBZ
	9	Reserved	
	8	MFX Sync Control Flag	
	7:6	Reserved	
Format:		MBZ	
5:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	
	Format:	=n	
	Total Length - 2		

MI_ARB_CHECK

MI_ARB_CHECK			
Source:	CommandStreamer		
Length Bias:	1		
Description			
This command allows software to enable or disable pre-fetch mechanism for command buffers in hardware.			
The command allows software to add preemption points in the ring buffer. The command streamer will preempt in the case arbitration is enabled, there is a pending execution list and this command is currently being parsed.			
Programming Notes			
MI_ARB_CHK command can be programmed in a ring buffer or batch buffer.			
MI_ARB_CHK command must not be programmed in INDIRECT_CTX and BB_PER_CTX_PTR buffers.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	05h MI_ARB_CHECK
		Format:	OpCode
	22:16	Reserved	
		Format:	MBZ
	15:8	Mask Bits	
		Programming Notes	
		Must be set to modify corresponding bits in Bits 7:0. (For implemented bits)	
7:1	Reserved		
	Format:	MBZ	
0	Pre-Parser Disable		
	This command allows software to enable or disable pre-parser of command buffer functionality from within a command sequence on per context basis. This ability allows the command stream to prefetch batch buffers that are yet to be parsed by looking ahead in the command FIFO. Even with this disabled, driver will have to ensure it does not self-modify commands already prefetched into the command buffer within the same batch buffer.		
	Value	Name	Description
	1		When set early fetch and parsing of future command buffers is disabled in hardware.
0		When reset early fetch and parsing of future command buffers is enabled in hardware when "Pre-Fetch Disable" in GFX_MODE register is not set.	

MI_ARB_CHECK	
	Programming Notes
	Mask bit [8] must be set to modify this bit. By default pre-fetch in hardware is enabled. The status of this bit is engine context save/restored.
	This is not a preemptible command if the corresponding mask bit is set for this field.

MI_ARB_ON_OFF

MI_ARB_ON_OFF			
Source:	CommandStreamer		
Length Bias:	1		
<p>The MI_ARB_ON_OFF instruction is used to disable/enable context switching. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.) This command should always be used as an off-on pair with the sequence of instructions to be protected from context switch between MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.</p>			
<p>The MI_ARB_ON_OFF instruction is used to disable/enable context switching. Context switching could be either due to preemption or un-successful wait for events or semaphore waits. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.)</p>			
Programming Notes			
<p>This command must be always be programmed in pairs of off/on in the same command dispatch. Sequence of instructions to be protected from context switch or preemption must be programmed between the MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.</p>			
<p>MI_ARB_ON_OFF command must not be programmed as part of the POSH command execution.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	08h MI_ARB_ON_OFF
		Format:	OpCode
	22:2	Reserved	
		Format:	MBZ
1	Arbitration Mode		
	Source:	RenderCS	

MI_ARB_ON_OFF

MI_ARB_ON_OFF									
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>Enable</td> </tr> </table> <p>This bit controls whether or not lite restore is allowed when arbitration is disabled thru clearing the Arbitration Enable bit. If arbitration is enabled then the value of this bit does not change the behavior of the hardware.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>Allow Lite Restore [Default]</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>Lite Restore Disabled</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	0h	Allow Lite Restore [Default]	1h	Lite Restore Disabled
	Format:	Enable							
	Value	Name							
	0h	Allow Lite Restore [Default]							
	1h	Lite Restore Disabled							
0	<p>Arbitration Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables or disables context switches due to pre-emption (a new execlist).</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Arbitration Enabled [Default]</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Arbitration Disabled</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	1	Arbitration Enabled [Default]	0	Arbitration Disabled
Format:	Enable								
Value	Name								
1	Arbitration Enabled [Default]								
0	Arbitration Disabled								

MI_ATOMIC

MI_ATOMIC											
Source:	BSpec										
Length Bias:	2										
Description											
<p>MI_ATOMIC is used to carry atomic operation on data in graphics memory. Atomic operations are supported on data granularity of 4B, 8B and 16B. The atomic operation leads to a read-modify-write operation on the data in graphics memory with the option of returning value. The data in graphics memory is modified by doing arithmetic and logical operation with the inline/indirect data provided with the MI_ATOMIC command. Inline/Indirect provided in the command can be one or two operands based on the atomic operation. Ex: Atomic-Compare operation needs two operands while Atomic-Add operation needs single operand and Atomic-increment requires no operand. Refer "Atomics" sub-section of "L3 Cache and URB" section of the B-spec for detailed atomic operations supported. Atomic operations can be enabled to return value by setting "Return Data Control" field in the command, return data is stored to CS_GPR registers. CS_GPR4/5 registers are updated with memory Return Data based on the "Data Size". Each GPR register is qword in size and occupies two MMIO registers.</p> <p>Note: Any references to CS_GPR registers in the command should be understood as the CS_GPR registers belonging to the corresponding engines *CS_GPR registers.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Engine Name</th> <th style="text-align: left;">Corresponding GPR Registers</th> </tr> </thead> <tbody> <tr> <td>RCS, POCS</td> <td>CS_GPR, POCS_GPR</td> </tr> <tr> <td>BCS</td> <td>BCS_GPR</td> </tr> <tr> <td>VCS</td> <td>VCS_GPR</td> </tr> <tr> <td>VECS</td> <td>VECS_GPR</td> </tr> </tbody> </table> <p>Indirect Source Operands: Operand1 is sourced from [CS_GPR1, CS_GPR0] Operand2 is sourced from [CS_GPR3, CS_GPR2] Read return Data is stored in [CS_GPR_5, CS_GPR4]</p> <p>When "Data Size" is DWORD lower dword of CS_GPR4 (Qword) is updated with the dword data returned from memory. When "Data Size" is QWORD only CS_GPR4 (Qword) is updated with the qword data returned from memory. When the data size is OCTWORD CS_GPR4/5 are updated with the OCTWORD data returned from memory. CS_GPR4 is loaded with lower qword returned from memory and CS_GPR5 is loaded with upper qword returned from memory.</p>		Engine Name	Corresponding GPR Registers	RCS, POCS	CS_GPR, POCS_GPR	BCS	BCS_GPR	VCS	VCS_GPR	VECS	VECS_GPR
Engine Name	Corresponding GPR Registers										
RCS, POCS	CS_GPR, POCS_GPR										
BCS	BCS_GPR										
VCS	VCS_GPR										
VECS	VECS_GPR										
Programming Notes											
<ul style="list-style-type: none"> • When Inline Data mode is not set, Dwords 3..10 must not be included as part of the command. Dword Length field in the header must be programmed accordingly. • When Inline Data Mode is set, Dwords3..10 must be included based on the Data Size field of the header. Both Operand-1 and Operand-2 dwords must be programmed based on the Data Size field. Operand-2 must be programmed to 0x0 if the atomic operation doesn't require it. Dword Length field in the header must be programmed accordingly. 											

DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	0h MI_COMMAND	
		Format:	OpCode	
	28:23	MI Command Opcode		
		Default Value:	2Fh MI_ATOMIC	
		Format:	OpCode	
	22	Memory Type		
		This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.		
		Value	Name	Description
		0h	Per Process Graphics Address	
1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.		
21	Post-Sync Operation			
	Source:	RenderCS, PositionCS		
	Value	Name	Description	
	0h	No Post Sync Operation	Command is executed as usual.	
	1h	Post Sync Operation	MI_ATOMIC command is executed as a pipelined PIPE_CONTROL flush command with Atomics operation as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any outstanding flushes issued prior to this command. When this bit set following restriction apply to atomic operation: <ul style="list-style-type: none"> • Non-Compare atomic operations are supported on data granularity of 4B and 8B. DW3 is the lower dword of the operand and DW4 is the upper dword of the operand for the atomic operation. • Compare atomic operations are supported on data granularity of 4B. DW3 is Operand-0 and DW4 is Operand-1 for the atomic operation. • Atomic operations to GGTT/PPGTT memory surface are supported. • Only Inline data mode for atomic operand is supported, no support for indirect data mode. • No support for Return Data Control functionality. • No support for atomic operations on data granularity of 16B. 	

MI_ATOMIC

					<ul style="list-style-type: none"> No support for compare atomic operations on data granularity of 8B.
Programming Notes					
Any desired pipeline flush operation can be achieved by programming PIPE_CONTROL command prior to this command.					
When this bit is set Command Streamer sends a flush down the pipe and the atomic operation is saved as post sync operation. Command streamer goes on executing the following commands. Atomic operation saved as post sync operation is executed at some point later on completion of corresponding flush issued.					
When this bit is set atomic semaphore signal operation will be out of order with rest of the MI commands programmed in the ring buffer or batch buffer, it will be in order with respect to the post sync operations resulting due to PIPE_CONTROL command.					
20:19	Data Size This field indicates the size of the operand in dword/qword/octword on which atomic operation will be performed. Data size must match with the Atomic Opcode. Operation Data size could be 4B, 8B or 16B				
	Value	Name	Description		
	0h	DWORD	Operand size used by Atomic Operation is DWORD.		
	1h	QWORD	Operand Size used by Atomic Operation is QWORD.		
	2h	OCTWORD	Operand Size used by Atomic Operation is OCTWORD.		
	3h	RESERVED			
18	Inline Data This bit when set indicates the source operands are provided in line within the command. When reset the source operands are in CS_GPR registers.				
Programming Notes					
CS_GPR registers must be programmed with appropriate values before issuing MI_ATOMIC command with this field reset.					
17	CS STALL This bit when set command stream waits for completion of this command before executing the next command.				
Programming Notes					
Render Command Streamer Only: CS will not guarantee atomic operation to be complete upon setting this bit along with Post Sync Operation set. When Post Sync Operation is set, this bit has no significance.					
16	Return Data Control				
	Source:	RenderCS, BlitterCS, VideoCS, VideoEnhancementCS			
When Return Data Control is set the read return feature will be enabled during the atomic operation. Data is stored in CS_GPR5/4 registers unconditionally on completion of the atomic operation. On data return CS_GPR5/4 Registers are updated based on the "Data Size" field. When					

MI_ATOMIC

		<p>"Data Size" is DWORD lower dword of CS_GPR4 (Qword) is updated with the dword data returned from memory. When "Data Size" is QWORD only CS_GPR4 (Qword) is updated with the qword data returned from memory. When the data size is OCTWORD CS_GPR4/5 are updated with the OCTWORD data returned from memory. CS_GPR4 is loaded with lower qword returned from memory and CS_GPR5 is loaded with upper qword returned from memory.</p>											
	15:8	<p>ATOMIC OPCODE This field selects the kind of atomic operation to be performed. Refer "Atomics" sub-section of "L3 Cache and URB" section for atomic opcode encoding and operation.</p>											
	7:0	<p>DWord Length</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>Total Length - 2. Excludes DWord (0,1).</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 50%;">Name</th> <th style="width: 30%;">Exists If</th> </tr> </thead> <tbody> <tr> <td>1h</td> <td>Inline Data 0 [Default]</td> <td>((Inline Data)==0)</td> </tr> <tr> <td>9h</td> <td>Inline Data 1</td> <td>((Inline Data)==1)</td> </tr> </tbody> </table>	Format:	=n	Value	Name	Exists If	1h	Inline Data 0 [Default]	((Inline Data)==0)	9h	Inline Data 1	((Inline Data)==1)
Format:	=n												
Value	Name	Exists If											
1h	Inline Data 0 [Default]	((Inline Data)==0)											
9h	Inline Data 1	((Inline Data)==1)											
1	31:2	<p>Memory Address</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>GraphicsAddress[31:2]</td> </tr> </table> <p>This field contains the graphics memory address of the data on which atomic operation has to be performed. Atomic operation can be performed on data granularity of 4B, 8B or 16B and hence the Address has to be correspondingly aligned to 4B,8B or 16B respectively.</p>	Format:	GraphicsAddress[31:2]									
	Format:	GraphicsAddress[31:2]											
	1	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
2	31:16	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
	Format:	MBZ											
15:0	<p>Memory Address High This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.</p>												
3	31:0	<p>Operand1 Data Dword 0</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>Dword0 of Operand1 when Inline Data mode is set.</p>	Format:	U32									
Format:	U32												
4	31:0	<p>Operand2 Data Dword 0</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>Dword0 of Operand2 when Inline Data mode is set.</p>	Format:	U32									
Format:	U32												
5	31:0	<p>Operand1 Data Dword 1</p>											

MI_ATOMIC				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U32</td> </tr> </table> <p>Dword1 of Operand1 when Inline Data mode is set.</p>	Format:	U32
Format:	U32			
6	31:0	<p>Operand2 Data Dword 1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U32</td> </tr> </table> <p>Dword1 of Operand2 when Inline Data mode is set.</p>	Format:	U32
Format:	U32			
7	31:0	<p>Operand1 Data Dword 2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U32</td> </tr> </table> <p>Dword2 of Operand1 when Inline Data mode is set.</p>	Format:	U32
Format:	U32			
8	31:0	<p>Operand2 Data Dword 2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U32</td> </tr> </table> <p>Dword2 of Operand2 when Inline Data mode is set.</p>	Format:	U32
Format:	U32			
9	31:0	<p>Operand1 Data Dword 3</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U32</td> </tr> </table> <p>Dword3 of Operand1 when Inline Data mode is set.</p>	Format:	U32
Format:	U32			
10	31:0	<p>Operand2 Data Dword 3</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">U32</td> </tr> </table> <p>Dword3 of Operand2 when Inline Data mode is set.</p>	Format:	U32
Format:	U32			

MI_BATCH_BUFFER_END

MI_BATCH_BUFFER_END			
Source:	CommandStreamer		
Length Bias:	1		
The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.			
Programming Notes			
SW must ensure it buffers the size of the command buffer beyond this command. The command buffer for the command streamer is 0.5KB.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0Ah MI_BATCH_BUFFER_END
		Format:	OpCode
	22:16	Reserved	
		Format:	MBZ
	15	Reserved	
		Format:	MBZ
14:1	Reserved		
	Format:	MBZ	
0	End Context		
	Format:	Enable	
This bit must only be set within a context image. If this command is parsed with this bit set then the engine will consider the context image restore complete. This bit is ignored if parsed during a batch buffer.			

MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START			
Source:	CommandStreamer		
Length Bias:	2		
<p>The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a batch buffer. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of <i>MI Functions</i>. The batch buffer can be specified as privileged or non-privileged, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of <i>MI Functions</i>.</p>			
Programming Notes			
<ul style="list-style-type: none"> A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command. It is essential that the address location beyond the current page be populated inside the GTT. HW performs over-fetch of the command addresses and any over-fetch requires a valid TLB entry. A single extra page beyond the batch buffer is sufficient. 			
<p>SW must ensure it buffers the size of the batch buffer includes additional buffer equal to the command buffer beyond the end. The command buffer for the command streamer is 0.5KB.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	31h MI_BATCH_BUFFER_START
		Format:	OpCode
	22	Second Level Batch Buffer	
		Exists If:	//MI_MODE:NestedBatchBufferEnable='0'
		<p>The command streamer contains three storage elements; one for the ring head address, one for the batch head address, and one for the second level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the first level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Chaining of second level batch buffers is supported. A chained second level batch buffer is inferred in hardware when a MI_BATCH_BUFFER_START command with "Second Level Batch Buffer" bit field set is executed from a second level batch buffer, hardware simply updates the head pointer of the second level batch address storage. Upon MI_BATCH_BUFFER_END, it will automatically return to the first level batch buffer address. This allows hardware to mimic a simple 3-level stack.</p>	
Value		Name	Description
0h		First level	Place the batch buffer address in the 1st (traditional) level

MI_BATCH_BUFFER_START

			batch	batch address storage element.
	1h	Second level batch		Place the batch buffer address in the second-level batch address storage element.
Programming Notes				
This field is ignored when MI_BATCH_BUFFER_START is executed from a ring. Whether this is a zero or one, the command streamer will move to the first level batch buffer.				
22	Nested Level Batch Buffer			
	Exists If:	//MI_MODE:NestedBatchBufferEnable= '1'		
Description				
<p>If this bit is set, the command streamer will move to the next level of batch buffer. Once it executes a MI_BATCH_BUFFER_END in the next level, it will return to the batch buffer executing this command and execute the next command. If clear then it will remain in the same batch buffer level and on executing MI_BATCH_BUFFER_END, it will return to the previous level. Otherwise known as batch buffer chaining.</p> <p>Hardware supports three levels of nesting, namely First Level, Second Level and Third Level.</p> <p>This bit must not be set in any of the MI_BATCH_BUFFER_START commands programmed as part of the 3rd level batch buffer's command sequence.</p>				
Value	Name	Description		
0h	Chain	Stay in the same batch buffer level.		
1h	Nested	Move to the next level of batch buffer.		
Programming Notes				
This field is ignored when MI_BATCH_BUFFER_START is executed from a ring. Whether this is a zero or one, the command streamer will move to the first level batch buffer.				
<p>Following programming guidelines must be follow for programming commands in third level batch buffer:</p> <ul style="list-style-type: none"> • Preemptable commands must not be programmed inside third level batch buffer. Refer section Preemption/Execlist Scheduling (Preemptable Commands) for the list of preemptable commands supported on per engine basis. Preemptable commands for RenderCS are mentioned below. <ul style="list-style-type: none"> • 3DPRIMITIVE and MI_ARB_CHK command in 3D mode of PIPELINE_SELECT operation. • GPGPU_WALKER, MEDIA_WALKER, MEDIA_OBJECT, PIPE_CONTROL, MI_ATOMIC (bit 21 set)and MEDIA_STATE_FLUSH in GPGPU or MEDIA mode of PIPELINE_SELECT operation. 				

MI_BATCH_BUFFER_START

- Any Non-Pipeline state command in GPGPU mode of PIPELINE_SELECT operation.
- MI_SEMAPHORE_WAIT and MI_WAIT_FOR_EVENT commands must not be programmed inside third level batch buffers. These commands are preemptable and also can result in context switch and hence must not be programmed inside third level batch buffer.

21 **POSH Start**

Source:	RenderCS, PositionCS
---------	----------------------

Batch buffers dedicated to be executed by POSH pipe are indicated by setting the field POSH Start in the MI_BATCH_BUFFER_START command header. Once POSH Start is set in a batch buffer all the following chained batch buffers and next level batch buffers will implicitly inherit the POSH Start field value. Once POSH Start is set in a batch buffer all the following command sequences are to be **executed** by POCS until the corresponding batch buffer sequencing is terminated through MI_BATCH_BUFFER_END/MI_CONDITIONAL_BATCH_BUFFER_END command.

Example:

- Once POSH Start is encountered in a first level batch buffer by POCS, it will get reset only when the first level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the ring buffer,
- Similarly, once POSH Start is encountered in a second level batch buffer by POCS, it will get reset only when the second level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the first level buffer,
- Similarly, once when POSH Start is encountered in a third level batch buffer by POCS, it will get reset only when the third level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the second level batch buffer.

Command sequences executed from the POSH Start batch buffer may lead to chained batch buffers or next level batch buffers. Batch buffers executed by POCS may have MI Commands, 3DSATE commands and 3DPRIMITIVE commands for POSH pipe. RCS on parsing MI_BATCH_BUFFER_START command with POSH Start enabled NOOPS the command and moves on the following command.

Value	Name	Description
0	[Default]	Batch buffer is not "POSH Start" enabled.
1		Batch buffer is "POSH Start" enabled.

Programming Notes

Resource Streamer must not be enabled for "POSH Enable" or "POSH Start" batch buffers.

20 **POSH Enable**

MI_BATCH_BUFFER_START

Source:	RenderCS, PositionCS
---------	----------------------

POSH Enable field in the MI_BATCH_BUFFER_START command is a hint to POCS to traverse (parse, don't execute) the batch buffer to look for POSH Start batch buffers. POSH Enable field is only inherited to the chained batch buffer and doesn't get inherit to the next level batch buffers unlike POSH Start field. POSH Enable field must be explicitly set in the MI_BATCH_BUFFER_START command which calls the next level batch buffers in order for the POCS to parse them to look for POSH Start batch buffers. POSH Start field takes precedence over the POSH Enable field in POCS.

Example:

- Once POSH Enable is encountered in a first level batch buffer, POCS will traverse the whole of the first level batch buffers (including chained first level) to check for POSH Start field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the second level batch buffers. SW must explicitly set the POSH Enable field for the second level batch buffer called from first level batch buffer if the second level batch buffer have to be traversed by POCS.
- Similarly, Once POSH Enable is encountered in a second level batch buffer, POCS will traverse the whole of the second level batch buffers (including chained second level) to check for POSH Start field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the third level batch buffers. SW must explicitly set the POSH Enable field for the third level batch buffer called from second level batch buffer if the third level batch buffer have to be traversed by POCS.
- Similarly, Once POSH Enable is encountered in a third level batch buffer, POCS will traverse the whole of the third level batch buffers (including chained second level) to check for POSH Start field in MI_BATCH_BUFFER_START command.

RCS ignores POSH Enable field and has no implications due to the POSH Enable field set in the MI_BATCH_BUFFER_START command.

Value	Name	Description
0	[Default]	Batch buffer is not "POSH Enable".
1		Batch buffer is "POSH Enable"/

Programming Notes

POCS executes only the MI_BATCH_BUFFER_START commands programmed in the ring buffer with POSH Enable set and NOOPS (predicates) all the other commands in the ring buffer. POCS only parses/traverses the batch buffer with POSH Enable to check for any batch buffer programmed with POSH Start set.

SW must set POSH Enable field in the MI_BATCH_BUFFER_START command programmed in ring buffer if the commands in the corresponding batch buffer or the chained batch buffers (includes Second Level and third level) has at least one batch buffer start command with POSH Start set (also implies 3DPRIMITIVE command for which POSH is enabled).

Resource Streamer must not be enabled for "POSH Enable" or "POSH Start" batch

MI_BATCH_BUFFER_START			
	buffers.		
19	<p>Enable Command Cache</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Source:</td> <td>RenderCS</td> </tr> </table> <p style="text-align: center;">Description</p> <p>Command Buffer DMA engine on processing the MI_BATCH_BUFFER_START command with Enable Command Cache set will make the corresponding command buffer read requests cacheable in L3, it also applies the command caching to all subsequent chained and next level batch buffers. Command buffer DMA engine uses the Command Buffer Cache Size programmed in the CMD_BUF_CCTL register to limit the read requests of a cacheable batch buffer to be cached in L3. DMA engine does this by tracking the amount of read requests made cacheable and stops caching when the read requested data size equals to the size of the command cache allocated. This avoids thrashing of cache for Batch Buffers larger in size compared to the command buffer cache allocated in L3. DMA engine resets the command caching tracker on events listed below.</p> <ul style="list-style-type: none"> • On an End Of Tile in PTRBR/POSH mode of operation. • On a context save of a context. • On command cache invalidation through PIPE_CONTROL. <p>Command buffer caching must not be enabled for batch buffers in GGTT address space.</p> <p>For best performance in terms of utilizing the command buffer section of L3 cache for PTBR mode, the recommendation is to ensure command buffers not to cross 128KB boundary. Avoiding a large overlap around 128KB boundary will minimize the cross distribution of 1KB chunks randomly between the L3 banks creating hotspots otherwise.</p> <p>Command Buffer DMA engine on processing the MI_BATCH_BUFFER_START command with Enable Command Cache set will make the corresponding command buffer read requests cacheable in L3, it also applies the command caching to all subsequent chained and next level batch buffers. Command buffer DMA engine uses the Command Buffer Cache Size programmed in the CMD_BUF_CCTL register to limit the read requests of a cacheable batch buffer to be cached in L3. DMA engine does this by tracking the amount of read requests made cacheable and stops caching when the read requested data size equals to the size of the command cache allocated. This avoids thrashing of cache for Batch Buffers larger in size compared to the command buffer cache allocated in L3. DMA engine resets the command caching tracker on events listed below.</p> <ul style="list-style-type: none"> • On an End Of Tile in PTRBR mode of operation. • On a context save of a context. • On command cache invalidation through PIPE_CONTROL. <p>Command buffer caching must not be enabled for batch buffers in GGTT address space.</p>	Source:	RenderCS
Source:	RenderCS		

MI_BATCH_BUFFER_START

For best performance in terms of utilizing the command buffer section of L3 cache for PTBR mode, the recommendation is to ensure command buffers not to cross 128KB boundary. Avoiding a large overlap around 128KB boundary will minimize the cross distribution of 1KB chunks randomly between the L3 banks creating hotspots otherwise.

Value	Name	Description
1		Command Cache enabled
0	[Default]	Command Cache disable

18	Reserved	
	Format:	MBZ

17:16	Reserved	
	Format:	MBZ

15	Predication Enable	
	Source:	RenderCS, PositionCS, BlitterCS, VideoCS, VideoEnhancementCS, ComputeCS
	<p>This bit is used to enable predication of this command. If this bit is set and Bit 0 of the MI_PREDICATE_RESULT_1 register is clear, this command is ignored. Otherwise the command is performed normally.</p>	

14:10	Reserved	
	Format:	MBZ

9	Reserved	
---	-----------------	--

8	Address Space Indicator	
	Description	
	<p>Batch buffers accessed via PPGTT are considered as non-privileged. Certain operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain.</p> <ul style="list-style-type: none"> • Chained or Second level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT. • Chained or Second level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware. <p>Batch buffers accessed via PPGTT are considered as non-privileged. Certain operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from</p>	

MI_BATCH_BUFFER_START

		<p>within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain.</p> <ul style="list-style-type: none"> • Chained or Nestedlevel batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT. • Chained or Nestedlevel batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware. 		
		Value	Name	Description
		0h	GGTT	This batch buffer is located in GGTT memory and is privileged.
		1h	PPGTT	This batch buffer is located in PPGTT memory and is Non-Privileged.
		Programming Notes		
		This field must be '0' unless the Per-Process GTT Enable is '1'		
	7:0	DWord Length		
		Default Value:	1h Excludes DWord (0,1)	
		Format:	=n	
		Total - Bias. Excludes DWord (0,1).		
<p>1..2 The address is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper 63 down to 48 bits are reserved bits which are ignored.</p>	<p>63:2</p>	Batch Buffer Start Address		
		Format:	VIRTUAL_ADDR[63:2]	
	<p>1:0</p>	Reserved		
		Format:	MBZ	

MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END			
Source:	CommandStreamer		
Length Bias:	2		
Description			
<p>The MI_CONDITIONAL_BATCH_BUFFER_END command is used to conditionally terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.</p> <p>Termination of the current level of batch buffer from which MI_CONDITIONAL_BATCH_BUFFER_END is executed or termination of all levels of batch buffer behavior is controlled by the End Current Batch Buffer Level bit in the command header.</p>			
Programming Notes			
<p>Any updates to the memory location exercised by this command must be ensured to be coherent in memory prior to programming of this command. If the memory location is being updated by a prior executed MI command (ex: MI_STORE_REGISTER_MEM ..etc) on the same engine, SW must follow one of the below programming note prior to programming of this command to ensure data is coherent in memory.</p> <p>Option1: Programming of "4" dummy MI_STORE_DATA_IMM (write to scratch space) commands prior to programming of this command. Example: MI_STORE_REGISTER_MEM (0x2288, 0x2CF0_0000) MI_STORE_DATA_IMM (4 times) (Dummy data, Scratch Address) MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000)</p> <p>Option2: Programming of a PIPE_CONTROL with Post-Sync Operation selected to Write Immediate Data to scratch space address with Command Streamer Stall Enable set prior to programming of this command. Example: MI_STORE_REGISTER_MEM (0x2288, 0x2CF0_0000) PIPE_CONTROL (Stall, Write Immediate Data), MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000) .</p> <p>Option3: MI_ATOMIC (write to scratch space) with "CS STALL" set prior to programming of this command. Example: MI_STORE_REGISTER_MEM (0x2288, 0x2CF0_0000) MI_ATOMIC (MOV, Dummy data, Scratch Address), MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000) .</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	36h MI_CONDITIONAL_BATCH_BUFFER_END
		Format:	OpCode
	22	Use Global GTT	
		Default Value:	0h
		Format:	Boolean
<p>If set, this command will use the global GTT to translate the Compare Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the Compare Address.</p>			

MI_CONDITIONAL_BATCH_BUFFER_END

21	Compare Semaphore	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>Boolean</td> </tr> </table> <p>This bit provides a means to enable or disable compare operation.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1h</td> <td></td> <td>The data from the Compare Data Dword (inline) is compared to the data in memory pointed by the Compare Address as per the Compare Operation. Based on the outcome of the compare operation it may result in either continue execution of the batch buffer or it may result in termination of the batch buffer.</td> </tr> <tr> <td>0h</td> <td></td> <td>This command will be treated as NOOP and usual batch buffer execution flow continues.</td> </tr> </tbody> </table>	Format:	Boolean	Value	Name	Description	1h		The data from the Compare Data Dword (inline) is compared to the data in memory pointed by the Compare Address as per the Compare Operation. Based on the outcome of the compare operation it may result in either continue execution of the batch buffer or it may result in termination of the batch buffer.	0h		This command will be treated as NOOP and usual batch buffer execution flow continues.
Format:	Boolean												
Value	Name	Description											
1h		The data from the Compare Data Dword (inline) is compared to the data in memory pointed by the Compare Address as per the Compare Operation. Based on the outcome of the compare operation it may result in either continue execution of the batch buffer or it may result in termination of the batch buffer.											
0h		This command will be treated as NOOP and usual batch buffer execution flow continues.											
20	Reserved												
19	Compare Mask Mode	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Compare Mask Mode Disabled</td> <td>Compare address points to Dword in memory consisting of Data Dword(DW0). HW will compare Data Dword(DW0) against Semaphore Data Dword.</td> </tr> <tr> <td>1h</td> <td>Compare Mask Mode Enabled</td> <td>Compare address points to Qword in memory consisting of compare Mask (DW0) and Data Dword(DW1). HW will do AND operation on Mask(DW0) with Data Dword(DW1) and then compare the result against Semaphore Data Dword.</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr style="background-color: #e1eef6;"> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">When "Compare Mask Mode" is enabled, "Compare Address" must be qword aligned.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Compare Mask Mode Disabled	Compare address points to Dword in memory consisting of Data Dword(DW0). HW will compare Data Dword(DW0) against Semaphore Data Dword.	1h	Compare Mask Mode Enabled	Compare address points to Qword in memory consisting of compare Mask (DW0) and Data Dword(DW1). HW will do AND operation on Mask(DW0) with Data Dword(DW1) and then compare the result against Semaphore Data Dword.	Programming Notes	When "Compare Mask Mode" is enabled, "Compare Address" must be qword aligned.
Value	Name	Description											
0h	Compare Mask Mode Disabled	Compare address points to Dword in memory consisting of Data Dword(DW0). HW will compare Data Dword(DW0) against Semaphore Data Dword.											
1h	Compare Mask Mode Enabled	Compare address points to Qword in memory consisting of compare Mask (DW0) and Data Dword(DW1). HW will do AND operation on Mask(DW0) with Data Dword(DW1) and then compare the result against Semaphore Data Dword.											
Programming Notes													
When "Compare Mask Mode" is enabled, "Compare Address" must be qword aligned.													
18	End Current Batch Buffer Level	<p>This field specifies if the current level of the batch buffer execution must or the complete batch (including parent) buffer execution must be terminated on compare operation evaluating false.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td> <p>Execution of the command results in terminating the batch buffer level from which this command has been executed and command execution returns to the previous/parent batch buffer. Ex:</p> <ul style="list-style-type: none"> when executed from a first level batch buffer, execution of batch buffer is terminated and the command execution resumes to the ring buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from first level batch buffer. </td> </tr> </tbody> </table>	Value	Name	Description	1		<p>Execution of the command results in terminating the batch buffer level from which this command has been executed and command execution returns to the previous/parent batch buffer. Ex:</p> <ul style="list-style-type: none"> when executed from a first level batch buffer, execution of batch buffer is terminated and the command execution resumes to the ring buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from first level batch buffer. 					
Value	Name	Description											
1		<p>Execution of the command results in terminating the batch buffer level from which this command has been executed and command execution returns to the previous/parent batch buffer. Ex:</p> <ul style="list-style-type: none"> when executed from a first level batch buffer, execution of batch buffer is terminated and the command execution resumes to the ring buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from first level batch buffer. 											

MI_CONDITIONAL_BATCH_BUFFER_END

				<ul style="list-style-type: none"> when executed from a second level batch buffer, execution of second level batch buffer is terminated and the command execution resumes to the first level batch buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from second level batch buffer. when executed from a third level batch buffer (if supported), execution of third level batch buffer is terminated and the command execution resumes to the second level batch buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from third level batch buffer. 																											
	0			Execution of the command result in termination of all levels of batch buffer and command execution returns to the ring buffer.																											
	17:16	Reserved																													
		Format:		MBZ																											
	15	Reserved																													
		Format:		MBZ																											
	14:12	Compare Operation																													
		<p>This field specifies the operation that will be executed to create the result that will either allow to continue or terminate the batch buffer.</p> <p>MAD = Memory Address Data Dword IDD = Inline Data Dword</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td>MAD_GREATER_THAN_IDD</td> <td>If Indirect fetched data is greater than inline data then continue.</td> </tr> <tr> <td style="text-align: center;">1h</td> <td>MAD_GREATER_THAN_OR_EQUAL_IDD</td> <td>If Indirect fetched data is greater than or equal to inline data then continue.</td> </tr> <tr> <td style="text-align: center;">2h</td> <td>MAD_LESS_THAN_IDD</td> <td>If Indirect fetched data is less than inline data then continue.</td> </tr> <tr> <td style="text-align: center;">3h</td> <td>MAD_LESS_THAN_OR_EQUAL_IDD</td> <td>If Indirect fetched data is less than or equal to inline data then continue.</td> </tr> <tr> <td style="text-align: center;">4h</td> <td>MAD_EQUAL_IDD</td> <td>If Indirect fetched data is equal to inline data then continue.</td> </tr> <tr> <td style="text-align: center;">5h</td> <td>MAD_NOT_EQUAL_IDD</td> <td>If Indirect fetched data is not equal to inline data then continue.</td> </tr> <tr> <td style="text-align: center;">6h</td> <td>Reserved</td> <td></td> </tr> <tr> <td style="text-align: center;">7h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>			Value	Name	Description	0h	MAD_GREATER_THAN_IDD	If Indirect fetched data is greater than inline data then continue.	1h	MAD_GREATER_THAN_OR_EQUAL_IDD	If Indirect fetched data is greater than or equal to inline data then continue.	2h	MAD_LESS_THAN_IDD	If Indirect fetched data is less than inline data then continue.	3h	MAD_LESS_THAN_OR_EQUAL_IDD	If Indirect fetched data is less than or equal to inline data then continue.	4h	MAD_EQUAL_IDD	If Indirect fetched data is equal to inline data then continue.	5h	MAD_NOT_EQUAL_IDD	If Indirect fetched data is not equal to inline data then continue.	6h	Reserved		7h	Reserved	
Value	Name	Description																													
0h	MAD_GREATER_THAN_IDD	If Indirect fetched data is greater than inline data then continue.																													
1h	MAD_GREATER_THAN_OR_EQUAL_IDD	If Indirect fetched data is greater than or equal to inline data then continue.																													
2h	MAD_LESS_THAN_IDD	If Indirect fetched data is less than inline data then continue.																													
3h	MAD_LESS_THAN_OR_EQUAL_IDD	If Indirect fetched data is less than or equal to inline data then continue.																													
4h	MAD_EQUAL_IDD	If Indirect fetched data is equal to inline data then continue.																													
5h	MAD_NOT_EQUAL_IDD	If Indirect fetched data is not equal to inline data then continue.																													
6h	Reserved																														
7h	Reserved																														
	11:8	Reserved																													

MI_CONDITIONAL_BATCH_BUFFER_END						
		<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
	7:0	DWord Length <table border="1"> <tr> <td>Default Value:</td> <td>2h Excludes DWord (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	2h Excludes DWord (0,1)	Format:	=n
Default Value:	2h Excludes DWord (0,1)					
Format:	=n					
1	31:0	Compare Data Dword Data dword to compare memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at Compare Address is greater than this dword, the execution of the command buffer should continue.				
2..3	63:3	Compare Address <table border="1"> <tr> <td>Format:</td> <td>VIRTUAL_ADDR[63:3]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:3]		
Format:	VIRTUAL_ADDR[63:3]					
Qword address to fetch Data Qword from memory. This field specifies the 4GB aligned base address of Gfx 4GB virtual address space within the host's 64-bit virtual address space. Virtual Address is a 64-bit value [63:0], but only a portion of it is used by hardware. Upper reserved bits are ignored and MBZ.	2:0	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM				
Source:	BlitterCS			
Length Bias:	2			
<p>The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.</p>				
Programming Notes				
<p>This command should not be used within a "non_privilege"batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	0h MI_COMMAND	
		Format:	OpCode	
	28:23	MI Command Opcode		
		Default Value:	2Eh MI_COPY_MEM_MEM	
		Format:	OpCode	
	22	Use Global GTT Source		
		It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.		
		Value	Name	Description
		0h	Per Process Graphics Address	
1h		Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.	
21	Use Global GTT Destination			
	It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.			
	Value	Name	Description	
0h	Per Process Graphics			

MI_COPY_MEM_MEM			
		Address	
	1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	20:8	Reserved	
		Format:	MBZ
	7:0	DWord Length	
		Default Value:	3 Excludes DWord (0,1)
		Format:	=n
1..2 Surface Type: MMIO Register Virtual Address is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.	63:0	Destination Memory Address	
		Format:	VIRTUAL_ADDR[63:0]
3..4 Surface Type: MMIO Register Virtual Address is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.	63:0	Source Memory Address	
		Format:	VIRTUAL_ADDR[63:0]

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM				
Source:	RenderCS			
Length Bias:	2			
<p>The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.</p>				
Programming Notes				
<p>This command should not be used within a "non_privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	0h MI_COMMAND	
		Format:	OpCode	
	28:23	MI Command Opcode		
		Default Value:	2Eh MI_COPY_MEM_MEM	
		Format:	OpCode	
	22	Use Global GTT Source		
		It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.		
		Value	Name	Description
		0h	Per Process Graphics Address	
1h		Global Graphics Address	It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.	
21	Use Global GTT Destination			
	This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit <i>must</i> be '1' if the Per Process GTT Enable bit is clear. This bit will determine write to memory uses Global or Per Process GTT.			
	Value	Name	Description	
0h	Per Process Graphics Address			

MI_COPY_MEM_MEM			
	1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	20:8	Reserved	
		Format:	MBZ
	7:0	DWord Length	
		Default Value:	3 Excludes DWord (0,1)
		Format:	=n
1..2	63:2	Destination Memory Address	
		Format:	GraphicsAddress[63:2]
		Surface Type: MMIO Register This field specifies the address of the memory location where the value fetched specified in the DWord address above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].	
	1:0	Reserved	
		Format:	MBZ
3.4	63:2	Source Memory Address	
		Format:	GraphicsAddress[63:2]
		Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].	
	1:0	Reserved	
		Format:	MBZ

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM		
Source:	VideoCS	
Length Bias:	2	
<p>The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.</p>		
Programming Notes		
<p>This command should not be used within a "non-privileged" batch buffer to access global virtual space; doing so will be treated as privilege access violation. Refer to the "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to learn more about HW behavior on encountering a privilege access violation.</p> <p>This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND
	Format: OpCode	
	28:23	MI Command Opcode
Default Value: 2Eh MI_MEM_TO_MEM		
Format: OpCode		
22	Use Global GTT Source	
	It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.	
	Value	Name
	0h	Per Process Graphics Address
1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
21	Use Global GTT Destination	
	It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.	
	Value	Name
	0h	Per Process Graphics Address
1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch

MI_COPY_MEM_MEM		
		buffer.
	20:8	Reserved Format: MBZ
	7:0	DWord Length Default Value: 3 Excludes DWord (0,1) Format: =n
1..2	63:2	Destination Memory Address Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	Reserved Format: MBZ
3..4	63:2	Source Memory Address Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	Reserved Format: MBZ

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM		
Source:	VideoEnhancementCS	
Length Bias:	2	
<p>The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.</p>		
Programming Notes		
<p>This command should not be used within a "non-privileged" batch buffer to access global virtual space; doing so will be treated as privilege access violation. Refer to the "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to learn more about HW behavior on encountering a privilege access violation.</p> <p>This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND
	Format: OpCode	
	28:23	MI Command Opcode
Default Value: 2Eh MI_MEM_TO_MEM		
Format: OpCode		
22	Use Global GTT Source	
	It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.	
	Value	Name
	0h	Per Process Graphics Address
1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
21	Use Global GTT Destination	
	It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.	
	Value	Name
	0h	Per Process Graphics Address
1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch

MI_COPY_MEM_MEM		
		buffer.
	20:8	Reserved Format: MBZ
	7:0	DWord Length Default Value: 3 Excludes DWord (0,1) Format: =n
1..2	63:2	Destination Memory Address Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	Reserved Format: MBZ
3..4	63:2	Source Memory Address Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	Reserved Format: MBZ

MI_DISPLAY_FLIP

MI_DISPLAY_FLIP	
Source:	RenderCS, BlitterCS
Length Bias:	2
<p>The MI_DISPLAY_FLIP command is used to request a specific display plane to switch (flip) to display a new buffer. The buffer is specified with a starting address and pitch. The tiled attribute of the buffer start address is programmed as part of the packet.</p> <p>The operation this command performs is also known as a "display flip request" operation - in that the flip operation itself will occur at some point in the future. This command specifies when the flip operation is to occur: either synchronously with vertical retrace to avoid tearing artifacts (possibly on a future frame), or asynchronously (as soon as possible) to minimize rendering stalls at the cost of tearing artifacts.</p>	
Programming Notes	
<ol style="list-style-type: none"> 1. This command simply requests a display flip operation. Command execution then continues normally. There is no guarantee that the flip (even if asynchronous) will occur prior to subsequent commands being executed. (Note that completion of the MI_FLUSH command does not guarantee that outstanding flip operations have completed). The MI_WAIT_FOR_EVENT command can be used to provide this synchronization - by pausing command execution until a pending flip has actually completed. This synchronization can also be performed by use of the Display Flip Pending hardware status. See Display Flip Synchronization in the Device Programming Interface chapter of MI Functions. 2. After a display flip operation is requested, software is responsible for initiating any required synchronization with subsequent buffer clear or rendering operations. For multi-buffering (e.g., double buffering) operations, this will typically require updating SURFACE_STATE or the binding table to change the rendering (back) buffer. In addition, prior to any subsequent clear or rendering operations, software must typically ensure that the new rendering buffer is not actively being displayed. Again, the MI_WAIT_FOR_EVENT command or Display Flip Pending hardware status can be used to provide this synchronization. See Display Flip Synchronization in the Device Programming Interface chapter of MI Functions. 3. The display buffer command uses the X and Y offset for the tiled buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For tiled buffers, the display subsystem uses the X and Y offset in generation of the final request to memory. The offset is always updated on the next vblank for both Synchronous and Asynch Flips. It is not necessary to have a flip enqueued to update the X and Y offset. 4. The display buffer command uses the linear dword offset for the linear buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For linear buffers, the display subsystem uses the dword offset in generation of the final request to memory. <ul style="list-style-type: none"> • For synchronous flips the offset is updated on the next vblank. It is not necessary to have a sync flip enqueued to update the dword offset. 5. DWord 3 (Left Eye Display Buffer Base Address) must not be set with synchronous flips or asynchronous flips. It is only allowed to be sent with stereo 3D flips. 	
Asynchronous flip completion time depends greatly on how much data has been prefetched for power savings,	

MI_DISPLAY_FLIP

and can take up to 1 full frame to complete. For faster flip completion, disable FBC and render compression and allocate a small amount of data buffer for the plane.

"Command Streamer Plane Number" mapping to "Display Plane Name" are listed in dsisplay B-spec -"Plane capability and Interoperability".

DWord	Bit	Description	
0	31:29	Command Type	
		Default Value: 0h MI_COMMAND	
		Format: OpCode	
	28:23		MI Command Opcode
			Default Value: 14h MI_DISPLAY_FLIP
			Format: OpCode
	22		Async Flip Indicator
			Format: Enable This bit should always be set if DW2 [1:0] == '01' (async flip). This field is required due to HW limitations. This bit is used by the render pipe while DW2 is used by the display hardware.
	21:19		Reserved
			Format: MBZ
	18:17		Reserved
			Format: MBZ
	16:14		Reserved
Format: MBZ			
13:8		Display Plane Select	
		Value	Name
		0h	Display Plane 1
		1h	Display Plane 2
		2h	Display Plane 3
		3h	Reserved
		4h	Display Plane 4
		5h	Display Plane 5
		6h	Display Plane 6
		7h	Display Plane 7
		8h	Display Plane 8
		9h	Display Plane 9
		Ah	Display Plane 10
		Bh	Display Plane 11
Ch	Display Plane 12		
Dh	Display Plane 13		

MI_DISPLAY_FLIP

		Eh	Display Plane 14									
		Fh	Display Plane 15									
		10h	Display Plane 16									
		11h	Display Plane 17									
		12h	Display Plane 18									
		13h	Display Plane 19									
		14h	Display Plane 20									
		15h	Display Plane 21									
		16h	Display Plane 22									
		17h	Display Plane 23									
		18h	Display Plane 24									
		19h	Display Plane 25									
		1Ah	Display Plane 26									
		1Bh	Display Plane 27									
		1Ch	Display Plane 28									
		1Dh	Display Plane 29									
		1Eh	Display Plane 30									
		1Fh	Display Plane 31									
		20h	Display Plane 32									
		[21h, 3Fh]	Reserved									
	7:0	DWord Length Format: <input type="text"/> =n Total Length - 2. Excludes DWord (0,1). For Synchronous Flips and Asynchronous Flips, this field must be programmed to 1h for a total length of 3. For Stereo 3D Flips, this field must be programmed to 2h for a total length of 4.										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Exists If</th> </tr> </thead> <tbody> <tr> <td>1h</td> <td>[Default]</td> <td>(([Flip Type]!='Stereo 3D Flip')</td> </tr> <tr> <td>2h</td> <td>[Default]</td> <td>(([Flip Type]='Stereo 3D Flip')</td> </tr> </tbody> </table>		Value	Name	Exists If	1h	[Default]	(([Flip Type]!='Stereo 3D Flip')	2h	[Default]	(([Flip Type]='Stereo 3D Flip')
Value	Name	Exists If										
1h	[Default]	(([Flip Type]!='Stereo 3D Flip')										
2h	[Default]	(([Flip Type]='Stereo 3D Flip')										
1	31	Stereoscopic 3D Mode Default Value: <input type="text"/> 0h Format: <input type="text"/> Enable This bit must be set if the Extra Display Buffer Address is part of this command. This bit is used to notify the display there is an extra DW before processing the Display Flip.										
	30:16	Reserved Format: <input type="text"/> MBZ										
	15:6	Display Buffer Pitch										

MI_DISPLAY_FLIP																											
	<table border="1"> <tr> <td>Default Value:</td> <td>0h</td> </tr> <tr> <td>Format:</td> <td>U10</td> </tr> </table> <p><i>For Synchronous Flips and Stereo 3D Flips only, this field specifies the pitch of the new display buffer. For Asynchronous Flips, this parameter is programmed so that all the flips in a flip chain should maintain the same pitch as programmed with the last synchronous flip or stereo 3D flip or direct through MMIO. See the Display Plane Stride register for details.</i></p>	Default Value:	0h	Format:	U10																						
Default Value:	0h																										
Format:	U10																										
5:3	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																								
Format:	MBZ																										
2:0	Tile Parameter <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>For Asynchronous Flips, this parameter cannot be changed. All the flips in a flip chain should maintain the same tile parameter as programmed with the last synchronous flip or direct through MMIO.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Linear [Default]</td> <td>For Synchronous Flips Only</td> </tr> <tr> <td>1h</td> <td>Tiled X</td> <td></td> </tr> <tr> <td>2h-3h</td> <td>Reserved</td> <td></td> </tr> <tr> <td>4h</td> <td>Tiled Y Legacy (Y B)</td> <td></td> </tr> </tbody> </table>	Format:	Enable	Value	Name	Description	0h	Linear [Default]	For Synchronous Flips Only	1h	Tiled X		2h-3h	Reserved		4h	Tiled Y Legacy (Y B)										
Format:	Enable																										
Value	Name	Description																									
0h	Linear [Default]	For Synchronous Flips Only																									
1h	Tiled X																										
2h-3h	Reserved																										
4h	Tiled Y Legacy (Y B)																										
2	<table border="1"> <tr> <td>31:12</td> <td> Display Buffer Base Address <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:12]</td> </tr> </table> <p>This field specifies Bits 31:12 of the Graphics Address of the new display buffer. In stereo 3D mode this is the right eye base address. In non-stereo 3D mode this is the only base address. (Refer to the Display Address Start Address Register description in the Display Registers chapter).</p> <table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2"> <ul style="list-style-type: none"> The Display buffer must reside completely in Main Memory. This address is always translated via the global (rather than per-process) GTT. </td> </tr> </tbody> </table> </td> </tr> <tr> <td>11</td> <td> Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> </td> </tr> <tr> <td>10:7</td> <td> Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> </td> </tr> <tr> <td>6:4</td> <td> Reserved </td> </tr> <tr> <td>3</td> <td> Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> </td> </tr> <tr> <td>2</td> <td> Reserved </td> </tr> <tr> <td>1:0</td> <td> Flip Type This field specifies whether the flip operation should be performed asynchronously to vertical retrace. </td> </tr> </table>	31:12	Display Buffer Base Address <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:12]</td> </tr> </table> <p>This field specifies Bits 31:12 of the Graphics Address of the new display buffer. In stereo 3D mode this is the right eye base address. In non-stereo 3D mode this is the only base address. (Refer to the Display Address Start Address Register description in the Display Registers chapter).</p> <table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2"> <ul style="list-style-type: none"> The Display buffer must reside completely in Main Memory. This address is always translated via the global (rather than per-process) GTT. </td> </tr> </tbody> </table>	Format:	GraphicsAddress[31:12]	Programming Notes		<ul style="list-style-type: none"> The Display buffer must reside completely in Main Memory. This address is always translated via the global (rather than per-process) GTT. 		11	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	10:7	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	6:4	Reserved	3	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	2	Reserved	1:0	Flip Type This field specifies whether the flip operation should be performed asynchronously to vertical retrace.
31:12	Display Buffer Base Address <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:12]</td> </tr> </table> <p>This field specifies Bits 31:12 of the Graphics Address of the new display buffer. In stereo 3D mode this is the right eye base address. In non-stereo 3D mode this is the only base address. (Refer to the Display Address Start Address Register description in the Display Registers chapter).</p> <table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2"> <ul style="list-style-type: none"> The Display buffer must reside completely in Main Memory. This address is always translated via the global (rather than per-process) GTT. </td> </tr> </tbody> </table>	Format:	GraphicsAddress[31:12]	Programming Notes		<ul style="list-style-type: none"> The Display buffer must reside completely in Main Memory. This address is always translated via the global (rather than per-process) GTT. 																					
Format:	GraphicsAddress[31:12]																										
Programming Notes																											
<ul style="list-style-type: none"> The Display buffer must reside completely in Main Memory. This address is always translated via the global (rather than per-process) GTT. 																											
11	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																								
Format:	MBZ																										
10:7	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																								
Format:	MBZ																										
6:4	Reserved																										
3	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																								
Format:	MBZ																										
2	Reserved																										
1:0	Flip Type This field specifies whether the flip operation should be performed asynchronously to vertical retrace.																										

MI_DISPLAY_FLIP			
	Value	Name	Description
	00b	Sync Flip [Default]	The flip will occur during the vertical blanking interval - thus avoiding any tearing artifacts.
	01b	Async Flip	The flip will occur "as soon as possible" - and may exhibit tearing artifacts
	10b	Stereo 3D Flip	The flip will occur during the vertical blanking interval (left or right eye blank selectable through display MMIO register) - thus avoiding any tearing artifacts.
	11b	Reserved	
Programming Notes			
<ul style="list-style-type: none"> The Display Buffer Pitch and Tile parameter cannot be changed for asynchronous flips (i.e., the new buffer must have the same pitch/tile format as the previous buffer). Only display surface base address can be changed. For Async Flips the Buffers used must be 32KB aligned. Async flips are supported on primary or universal planes only. 			
<ul style="list-style-type: none"> For Stereo 3D flips, both left and right eye buffers must have the same pitch and tile format. 			
3	31:12	Left Eye Display Buffer Base Address	
		Format:	GraphicsAddress[31:12]
This field specifies Bits 31:12 of the Graphics Address of the new display buffer for the stereo 3D left eye. In non-stereo 3D mode this address is not used. (Refer to the Display Address Start Address Register description in the Display Registers chapter).			
Programming Notes			
<ul style="list-style-type: none"> The Display buffer must reside completely in Main Memory. This address is always translated via the global (rather than per-process) GTT. 			
	11:0	Reserved	
		Format:	MBZ

MI_FLUSH_DW

MI_FLUSH_DW						
Source:	BlitterCS					
Length Bias:	2					
<p>The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to: Flush any dirty data to memory. Invalidate the TLB cache inside the hardware</p> <p>Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).</p>						
DWord	Bit	Description				
0	31:29	Command Type <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>0h MI_COMMAND</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h MI_COMMAND	Format:	OpCode
	Default Value:	0h MI_COMMAND				
	Format:	OpCode				
	28:23	MI Command Opcode <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>26h MI_FLUSH_DW</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	26h MI_FLUSH_DW	Format:	OpCode
	Default Value:	26h MI_FLUSH_DW				
	Format:	OpCode				
22	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
21	Store Data Index <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>U1</td> </tr> </table> <p>This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).</p>	Format:	U1			
Format:	U1					
20:19	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
18	TLB Invalidate <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>U1</td> </tr> </table> <p>If ENABLED, all TLBs belonging to Blitter Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.</p> <p>If GFX_MODE (0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.</p>	Format:	U1			
Format:	U1					

MI_FLUSH_DW

	17	Reserved	
		Format:	MBZ
	16	Reserved	
		Format:	MBZ
	15:14	Post-Sync Operation	
		BitFieldDesc	
		Value	Name
		Description	
	0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
	1h	Write Immediate Data QWord	Write the QWord containing Immediate Data Low, High DWs to the Destination Address
	2h	Reserved	Reserved
	3h	Write TIMESTAMP Register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.
		Programming Notes	
		<p>If executed in a PPGTT (non-secure) batch buffer, the post-sync op is always inhibited. This command does not write anything out to memory.</p> <p>[For all other devices]: If executed in a non-secure batch buffer, the address given is in a PPGTT address space. If in a secure ring or batch, the address given is in GGTT space.</p>	
	13:10	Reserved	
		Format:	MBZ
	9	Flush LLC	
		Format:	Enable
		<p>If enabled, at the end of the current MI_FLUSH_DW the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.</p>	
	8	Notify Enable	
		Format:	U1
		<p>If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.</p>	

MI_FLUSH_DW												
	7:6	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ								
		MBZ										
5:0	DWord Length Default Value: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">3h DWORD_COUNT_n</td></tr></table> Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">=n</td></tr></table> Total Length - 2. Excludes DWord (0,1).		3h DWORD_COUNT_n		=n							
	3h DWORD_COUNT_n											
	=n											
1..2 This field specifies the destination address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size. GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address.	63:48	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ								
		MBZ										
	47:3	Destination Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">GraphicsAddress[47:3]</td></tr></table>		GraphicsAddress[47:3]								
		GraphicsAddress[47:3]										
2	Destination Address Type Defines the address space of the Destination Address. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">PPGTT</td> <td>Use PPGTT address space for DW write</td> </tr> <tr> <td style="text-align: center;">1h</td> <td style="text-align: center;">GGTT</td> <td>Use GGTT address space for DW write</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; color: #0070C0;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">Ignored if "No write" is the selected in Operation.</td> </tr> </tbody> </table>	Value	Name	Description	0h	PPGTT	Use PPGTT address space for DW write	1h	GGTT	Use GGTT address space for DW write	Programming Notes	Ignored if "No write" is the selected in Operation.
Value	Name	Description										
0h	PPGTT	Use PPGTT address space for DW write										
1h	GGTT	Use GGTT address space for DW write										
Programming Notes												
Ignored if "No write" is the selected in Operation.												
1:0	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="text-align: center;">MBZ</td></tr></table>		MBZ									
	MBZ											
3..4	63:0 Immediate Data This field specifies the DWord value to be written to the targeted location. Only valid when 15:14 in header is set to 1h. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; color: #0070C0;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'</td> </tr> </tbody> </table>	Programming Notes	To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'									
Programming Notes												
To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'												

MI_FLUSH_DW

MI_FLUSH_DW			
Source:	VideoCS		
Length Bias:	2		
<p>The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to: Flush any dirty data to memory. Invalidate the TLB cache inside the hardware Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	26h MI_FLUSH_DW
		Format:	OpCode
	22	Reserved	
	21	Store Data Index	
		Format:	U1
	<p>This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).</p>		
20:19	Reserved		
	Format:	MBZ	
18	TLB Invalidate		
	Format:	U1	
<p>If ENABLED, all TLBs belonging to Video Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.</p>			
17:16	Reserved		
	Format:	MBZ	
15:14	Post-Sync Operation		
	BitFieldDesc		
	Value	Name	Description
	0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
1h	Write	HW implicitly detects the Data size to be Qword or Dword to be written to	

MI_FLUSH_DW

			Immediate Data	memory based on the command dword length programmed . When Dword Length indicates Qword, Writes the QWord containing Immediate Data Low, High DWs to the Destination Address . When Dword Length indicates Dword, Writes the DWord containing Immediate Data Low to the Destination Address
		2h	Reserved	Reserved
		3h		Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.
Programming Notes				
If executed in a PPGTT (non-secure) batch buffer, the post-sync op is always inhibited. This command does not write anything to memory.				
13:10	Reserved			
	Format:		MBZ	
9	Flush LLC			
	Format:		Enable	
	If enabled, at the end of the current MI_FLUSH_DW the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.			
8	Notify Enable			
	Format:		U1	
	If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.			
7	Video Pipeline Cache invalidate			
	Format:		U1	
	Enable the invalidation of the video cache at the end of this flush			
6	Reserved			
	Format:		MBZ	
5:0	DWord Length			
	Format:		=n	
	Value	Name		
	2h	DWord		
	3h	QWord [Default]		
1..2	63:57	Reserved		
		Format:		MBZ
	56:48	Reserved		

MI_FLUSH_DW											
		Format: MBZ									
47:3	Address	Format: GraphicsAddress[47:3]U64 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by the hardware. Upper bits [63:48] are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] is a 4KB page address.									
2	Destination Address Type	Defines address space of Destination Address <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 70%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>PPGTT</td> <td>Use PPGTT address space for DW write</td> </tr> <tr> <td>1h</td> <td>GGTT</td> <td>Use GGTT address space for DW write</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> Ignored if "No write" is the selected in Operation.	Value	Name	Description	0h	PPGTT	Use PPGTT address space for DW write	1h	GGTT	Use GGTT address space for DW write
Value	Name	Description									
0h	PPGTT	Use PPGTT address space for DW write									
1h	GGTT	Use GGTT address space for DW write									
1:0	Reserved	Format: MBZ									
3..4	63:0	Immediate Data This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'									

MI_FLUSH_DW

MI_FLUSH_DW			
Source:	VideoEnhancementCS		
Length Bias:	2		
<p>The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to:</p> <ul style="list-style-type: none"> • Flush any dirty data to memory. • Invalidate the TLB cache inside the hardware <p>Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	26h MI_FLUSH_DW
		Format:	OpCode
	22	Reserved	
	21	Store Data Index	
		Format:	U1
	<p>This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).</p>		
20:19	Reserved		
	Format:	MBZ	
18	TLB Invalidate		
	Format:	U1	
	<p>If ENABLED, all TLBs belonging to Video Enhancement Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.</p>		
	<p>If GFX_MODE (0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.</p>		
17:16	Reserved		
	Format:	MBZ	

MI_FLUSH_DW																	
	15:14	<p>Post-Sync Operation</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>No Write</td> <td>No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.</td> </tr> <tr> <td>1h</td> <td>Write Immediate Data</td> <td>Write the QWord containing Immediate Data Low, High DWs to the Destination Address</td> </tr> <tr> <td>2h</td> <td>Reserved</td> <td>Reserved</td> </tr> <tr> <td>3h</td> <td>Write TIMESTAMP register</td> <td>Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>If executed in non-secure batch buffer, the address given will be in a PPGTT address space. If in a secure ring or batch, address given will be in GGTT space</p>	Value	Name	Description	0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.	1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address	2h	Reserved	Reserved	3h	Write TIMESTAMP register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.
	Value	Name	Description														
	0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.														
	1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address														
	2h	Reserved	Reserved														
	3h	Write TIMESTAMP register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.														
	13:10	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ													
	Format:	MBZ															
	9	<p>Flush LLC</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>If enabled, at the end of the current MI_FLUSH_DW the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.</p>	Format:	Enable													
	Format:	Enable															
8	<p>Notify Enable</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.</p>	Format:	U1														
Format:	U1																
7:6	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
5:0	<p>DWord Length</p> <table border="1"> <tr> <td>Default Value:</td> <td>3h Excludes DWord (0,1) = 2 for DWord, 3 for QWord</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	Default Value:	3h Excludes DWord (0,1) = 2 for DWord, 3 for QWord	Format:	=n												
Default Value:	3h Excludes DWord (0,1) = 2 for DWord, 3 for QWord																
Format:	=n																
1	31:3	<p>Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:3]U28</td> </tr> </table> <p>This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.</p>	Format:	GraphicsAddress[31:3]U28													
	Format:	GraphicsAddress[31:3]U28															
2	<p>Destination Address Type</p> <p>Defines address space of Destination Address</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>PPGTT</td> <td>Use PPGTT address space for DW write</td> </tr> </tbody> </table>	Value	Name	Description	0h	PPGTT	Use PPGTT address space for DW write										
Value	Name	Description															
0h	PPGTT	Use PPGTT address space for DW write															

MI_FLUSH_DW		
	1h	GGTT Use GGTT address space for DW write
	Programming Notes	
	Ignored if "No write" is the selected in Operation.	
	1:0	Reserved
	Format:	MBZ
2	31:16	Reserved
	Format:	MBZ
	15:0	Address High
	Format:	GraphicsAddress[47:32]U64
	This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space	
3..4	63:0	Immediate Data
	This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h	
	To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'	

MI_FORCE_WAKEUP

MI_FORCE_WAKEUP		
Source:	CommandStreamer	
Length Bias:	2	
<p>This command is used to communicate Force Wakeup request to PM unit. No functionality is performed by this command when none of the mask bits are set and is equivalent to NOOP. Example for usage model: VCS Ring Buffer: MI_FORCE_WAKEUP (Force Render Awake set to '1') MI_SEMPAHORE_SIGNAL (Signal context id 0xABC to Render Command Streamer) MI_FORCE_WAKEUP (Force Render Awake set to '0') MI_BATCH_BUFFER_START STATE Commands ... MI_FORCE_WAKEUP (Force Render Awake set to '1') MI_LOAD_REGISTER_IMMEDIATE (Load register 0x23XX in render command streamer with data 0xFF) MI_FORCE_WAKEUP (Force Render Awake set to '0') .. MI_BATCH_BUFFER_END</p>		
Programming Notes		
<p>This command must not be programmed in the command stream for Render Engine Command Streamer or Position Command Streamer or Compute Engine Command Streamer.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND Format: OpCode
	28:23	MI Command Opcode
		Default Value: 1Dh MI_FORCE_WAKEUP Format: OpCode
22:8	Reserved	
	Format: MBZ	
7:0	DWord Length	Default Value: 0h Format: =n
		Total Length - 2. Excludes DWord (0,1).
1	31:16	Mask Bits
		Format: Mask[15:0]
	Programming Notes	
		Must be set to modify corresponding Bits 15:0. (Mask bits must not be set for reserved bits).
15:10	Reserved	Format: MBZ
9	MFX Power Well Control	Format: U1
		This Bit controls whether or not the MFX Power Well is powered. When this bit is unmasked, all

MI_FORCE_WAKEUP									
	<p>force awake bits programming are ignored.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Enable Power Well</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Disable Power Well [Default]</td> </tr> </tbody> </table>	Value	Name	1	Enable Power Well	0	Disable Power Well [Default]		
Value	Name								
1	Enable Power Well								
0	Disable Power Well [Default]								
8	<p>HEVC Power Well Control</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>This Bit controls whether or not the HEVC Power Well is powered. When this bit is unmasked, all force awake bits programming are ignored.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Enable Power Well</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Disable Power Well [Default]</td> </tr> </tbody> </table>	Format:	U1	Value	Name	1	Enable Power Well	0	Disable Power Well [Default]
Format:	U1								
Value	Name								
1	Enable Power Well								
0	Disable Power Well [Default]								
7:5	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
4	<p>Force Media-Slice3 Awake</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>When set, Command Streamer sends message to PM to force awake the media engines in media slice 3, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Mask bit [20] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.</td> </tr> </tbody> </table>	Format:	U1	Programming Notes	Mask bit [20] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.				
Format:	U1								
Programming Notes									
Mask bit [20] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.									
3	<p>Force Media-Slice2 Awake</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>When set, Command Streamer sends message to PM to force awake the media engines in media slice 2, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Mask bit [19] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.</td> </tr> </tbody> </table>	Format:	U1	Programming Notes	Mask bit [19] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.				
Format:	U1								
Programming Notes									
Mask bit [19] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.									
2	<p>Force Media-Slice1 Awake</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>When set, Command Streamer sends message to PM to force awake the media engines in media slice 1, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset,</p>	Format:	U1						
Format:	U1								

MI_FORCE_WAKEUP															
	<p>command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Mask bit [18] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.</td> </tr> </table>	Programming Notes		Mask bit [18] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.											
Programming Notes															
Mask bit [18] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.															
1	<p>Force Render Awake</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Source:</td> <td>BlitterCS, VideoCS, VideoEnhancementCS</td> </tr> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Description</th> </tr> <tr> <td colspan="2">When set, Command Streamer sends message to PM to force awake render engine (next instructions require render engine or compute engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of render engine (next instructions do not require the render engine or compute engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.</td> </tr> </table> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Mask bit [17] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.</td> </tr> <tr> <td colspan="2">MI_FORCE_WAKEUP command programmed in RenderCS command buffer must not set "Force Render Awake" bit.</td> </tr> </table>	Source:	BlitterCS, VideoCS, VideoEnhancementCS	Format:	U1	Description		When set, Command Streamer sends message to PM to force awake render engine (next instructions require render engine or compute engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of render engine (next instructions do not require the render engine or compute engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.		Programming Notes		Mask bit [17] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.		MI_FORCE_WAKEUP command programmed in RenderCS command buffer must not set "Force Render Awake" bit.	
Source:	BlitterCS, VideoCS, VideoEnhancementCS														
Format:	U1														
Description															
When set, Command Streamer sends message to PM to force awake render engine (next instructions require render engine or compute engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of render engine (next instructions do not require the render engine or compute engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.															
Programming Notes															
Mask bit [17] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.															
MI_FORCE_WAKEUP command programmed in RenderCS command buffer must not set "Force Render Awake" bit.															
0	<p>Force Media-Slice0 Awake</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U1</td> </tr> </table> <p>When set, Command Streamer sends message to PM to force awake the media engines in media slice 0, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Mask bit [16] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.</td> </tr> </table>	Format:	U1	Programming Notes		Mask bit [16] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.									
Format:	U1														
Programming Notes															
Mask bit [16] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed.															



MI_LOAD_REGISTER_IMM

MI_LOAD_REGISTER_IMM				
Source:	CommandStreamer			
Length Bias:	2			
<p>The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range). Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundary and not destination. Any other address will ensure the destination is updated prior to parsing the next command</p>				
Programming Notes				
<p>Many MMIO bits require the engine to be IDLE prior to updating the value. Command streamer does not implicitly put in a pipeline flush in the cases a MMIO bit requires the engine to be IDLE. In the case there was a 3DPRIMITIVE command or GPGPU_WALKER command without any stalling PIPE_CONTROL, one must be inserted prior to a MI_LOAD_REGISTER_IMMEDIATE that is updating a bit that requires the engine to be IDLE.</p>				
<p>When executed from a non-privileged batch buffer, MMIO writes are only allowed to the registers listed in User Mode Non-Privileged Registers for the corresponding engine, any writes targeting the register not listed in the User Mode Non-Privileged Register will convert this command to a NOOP.</p>				
<p>The following addresses should NOT be used for LRIs:</p> <ol style="list-style-type: none"> 1. 0x8800 - 0x88FF 2. >= 0xC0000 <p>Limited LRI cycles to the Display Engine (0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each LRI.</p>				
<p>Programming an MMIO register is equivalent to programming a non-pipeline state to the hardware and hence an explicit stalling flush needs to be programmed prior to programming this command. However for certain MMIO registers based on their functionality doing an explicit stalling flush is exempted. Listed below are the exempted registers.</p> <ul style="list-style-type: none"> • 3DPRIM_END_OFFSET - Auto Draw End Offset • 3DPRIM_START_VERTEX - Load Indirect Start Vertex • 3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count • 3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count • 3DPRIM_START_INSTANCE - Load Indirect Start Instance • 3DPRIM_BASE_VERTEX - Load Indirect Base Vertex • 3DPRIM_XP0 - Load Indirect Extended Parameter 0 • 3DPRIM_XP1 - Load Indirect Extended Parameter 1 • 3DPRIM_XP2 - Load Indirect Extended Parameter 2 				
DWord	Bit	Description		
0	31:29	<p>Command Type</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td style="width: 50%;">0h MI_COMMAND</td> </tr> </table>	Default Value:	0h MI_COMMAND
Default Value:	0h MI_COMMAND			

MI_LOAD_REGISTER_IMM

	Format:	OpCode																																								
28:23	MI Command Opcode																																									
	Default Value:	22h MI_LOAD_REGISTER_IMM																																								
	Format:	OpCode																																								
22:20	Reserved																																									
	Format:	MBZ																																								
19	Add CS MMIO Start Offset																																									
	This bit controls the functionality of the "Register Address" field in the command.																																									
Value	Name	Description																																								
1		<p>"Register Address" field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. However, during context restore bits [11:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Refer "Register Access and User Mode Privilege Access" section to get the list of all the instances of the Command Streamers and their associated MMIO Start Offset's.</p> <p>/// Command executed from Ring Buffer or Batch Buffer Example: MI_LOAD_REGISTER_IMMEDIATE, ADD_CS_MMIO_START_OFFSET: true, Data:0xABCD, Register Address: 0x00_2000 The above command when executed on RenderCS will result in a write to MMIO offset 0x00_4000 (0x00_2000 + 0x00_2000) instead to 0x00_2000. Note that RenderCS MMIO start offset is 0x2000. For illustration table below shows the result of this command executed from few instances of the command streamers from different engines.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin: 10px 0;"> <thead> <tr> <th>S.No</th> <th>Command Streamer</th> <th>Command Streamer MMIO Base</th> <th>LRI Register Offset</th> <th>LRI Written Address</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>RenderCS</td> <td>0x2000</td> <td>0x2000</td> <td>0x00_4000</td> </tr> <tr> <td>2</td> <td>VideoCS0</td> <td>0x1C_0000</td> <td>0x2000</td> <td>0x1C_2000</td> </tr> <tr> <td>3</td> <td>VideoCS1</td> <td>0x1C_4000</td> <td>0x2000</td> <td>0x1C_6000</td> </tr> <tr> <td>4</td> <td>VideoEnhancement0</td> <td>0x1C_8000</td> <td>0x2000</td> <td>0x1C_A000</td> </tr> <tr> <td>5</td> <td>VideoCS2</td> <td>0x1D_0000</td> <td>0x2000</td> <td>0x1D_2000</td> </tr> <tr> <td>6</td> <td>VideoCS3</td> <td>0x1D_4000</td> <td>0x2000</td> <td>0x1D_6000</td> </tr> <tr> <td>7</td> <td>VideoEnhancement1</td> <td>0x1D_8000</td> <td>0x2000</td> <td>0x1D_A000</td> </tr> </tbody> </table> <p>// Command executed from context image during context restore Example: MI_LOAD_REGISTER_IMMEDIATE, ADD_CS_MMIO_START_OFFSET: true, Data:0xABCD, Register Address: 0x1C_2030 The above command when executed on RenderCS will result in a write to MMIO offset 0x2030 (0x00_2000 + 0x030) instead to 0x1C_2030. Note that</p>	S.No	Command Streamer	Command Streamer MMIO Base	LRI Register Offset	LRI Written Address	1	RenderCS	0x2000	0x2000	0x00_4000	2	VideoCS0	0x1C_0000	0x2000	0x1C_2000	3	VideoCS1	0x1C_4000	0x2000	0x1C_6000	4	VideoEnhancement0	0x1C_8000	0x2000	0x1C_A000	5	VideoCS2	0x1D_0000	0x2000	0x1D_2000	6	VideoCS3	0x1D_4000	0x2000	0x1D_6000	7	VideoEnhancement1	0x1D_8000	0x2000	0x1D_A000
S.No	Command Streamer	Command Streamer MMIO Base	LRI Register Offset	LRI Written Address																																						
1	RenderCS	0x2000	0x2000	0x00_4000																																						
2	VideoCS0	0x1C_0000	0x2000	0x1C_2000																																						
3	VideoCS1	0x1C_4000	0x2000	0x1C_6000																																						
4	VideoEnhancement0	0x1C_8000	0x2000	0x1C_A000																																						
5	VideoCS2	0x1D_0000	0x2000	0x1D_2000																																						
6	VideoCS3	0x1D_4000	0x2000	0x1D_6000																																						
7	VideoEnhancement1	0x1D_8000	0x2000	0x1D_A000																																						

MI_LOAD_REGISTER_IMM

			<p>RenderCS MMIO start offset is 0x2000. For illustration table below shows the result of this command executed from few instances of the command streamers from different engines during context restore.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 5%;">S.No</th> <th style="width: 25%;">Command Streamer</th> <th style="width: 25%;">Command Streamer MMIO Base</th> <th style="width: 15%;">LRI Register Offset</th> <th style="width: 30%;">LRI Written Address</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>RenderCS</td> <td>0x2000</td> <td>0x1C_2030</td> <td>0x00_2030</td> </tr> <tr> <td>2</td> <td>VideoCS0</td> <td>0x1C_0000</td> <td>0x1C_2030</td> <td>0x1C_0030</td> </tr> <tr> <td>3</td> <td>VideoCS1</td> <td>0x1C_4000</td> <td>0x1C_2030</td> <td>0x1C_4030</td> </tr> <tr> <td>4</td> <td>VideoEnhancement0</td> <td>0x1C_8000</td> <td>0x1C_2030</td> <td>0x1C_8030</td> </tr> <tr> <td>5</td> <td>VideoCS2</td> <td>0x1D_0000</td> <td>0x1C_2030</td> <td>0x1D_0030</td> </tr> <tr> <td>6</td> <td>VideoCS3</td> <td>0x1D_4000</td> <td>0x1C_2030</td> <td>0x1D_4030</td> </tr> <tr> <td>7</td> <td>VideoEnhancement1</td> <td>0x1D_8000</td> <td>0x1C_2030</td> <td>0x1D_8030</td> </tr> </tbody> </table>	S.No	Command Streamer	Command Streamer MMIO Base	LRI Register Offset	LRI Written Address	1	RenderCS	0x2000	0x1C_2030	0x00_2030	2	VideoCS0	0x1C_0000	0x1C_2030	0x1C_0030	3	VideoCS1	0x1C_4000	0x1C_2030	0x1C_4030	4	VideoEnhancement0	0x1C_8000	0x1C_2030	0x1C_8030	5	VideoCS2	0x1D_0000	0x1C_2030	0x1D_0030	6	VideoCS3	0x1D_4000	0x1C_2030	0x1D_4030	7	VideoEnhancement1	0x1D_8000	0x1C_2030	0x1D_8030
S.No	Command Streamer	Command Streamer MMIO Base	LRI Register Offset	LRI Written Address																																							
1	RenderCS	0x2000	0x1C_2030	0x00_2030																																							
2	VideoCS0	0x1C_0000	0x1C_2030	0x1C_0030																																							
3	VideoCS1	0x1C_4000	0x1C_2030	0x1C_4030																																							
4	VideoEnhancement0	0x1C_8000	0x1C_2030	0x1C_8030																																							
5	VideoCS2	0x1D_0000	0x1C_2030	0x1D_0030																																							
6	VideoCS3	0x1D_4000	0x1C_2030	0x1D_4030																																							
7	VideoEnhancement1	0x1D_8000	0x1C_2030	0x1D_8030																																							
		[Default]	"Register Address" field in the command is absolute and not an offset from the executing command streamer MMIO start offset..																																								
18	Reserved																																										
	Format:		MBZ																																								
17	<p>MMIO Remap Enable</p> <p>This bit provides a mechanism in HW to remap the MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.</p> <p>A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.</p> <p>This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td></td> <td>MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.</td> </tr> <tr> <td style="text-align: center;">0</td> <td></td> <td>MMIO remapping will not be applied to the MMIO address.</td> </tr> </tbody> </table>			Value	Name	Description	1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.	0		MMIO remapping will not be applied to the MMIO address.																															
Value	Name	Description																																									
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.																																									
0		MMIO remapping will not be applied to the MMIO address.																																									
	Programming Notes																																										
	<ul style="list-style-type: none"> SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands. MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks. 																																										

MI_LOAD_REGISTER_IMM	
	<ul style="list-style-type: none"> • "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa. • When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.
16:13	Reserved Format: MBZ
12	Reserved
11:8	Byte Write Disables Range: Must specify a valid register write operation If [11:8] is '1111b', then this command will behave as a NOOP. Otherwise, the value is forwarded to the destination register.
7:0	DWord Length Default Value: 1h Excludes DWord (0,1) Format: =n
1..2	63:32 Data DWord Mask: Bytes Write Disables Format: U32 This field specifies the DWord value to be written to the targeted location.
	31:23 Reserved Format: MBZ
	22:2 Register Offset Format: MmioAddress[22:2] This field specifies bits [22:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).
	1:0 Reserved Format: MBZ

MI_LOAD_REGISTER_MEM

MI_LOAD_REGISTER_MEM			
Source:	RenderCS, BlitterCS, VideoCS, VideoEnhancementCS		
Length Bias:	2		
The MI_LOAD_REGISTER_MEM command requests from a memory location and stores that DWord to a register.			
Programming Notes			
The command temporarily halts commands that will cause cycles down the 3D pipeline.			
The following addresses should NOT be used for MMIO writes: <ul style="list-style-type: none"> • 0x8800 - 0x88FF • >= 0xC0000 Limited MMIO writes cycles to the Display Engine (0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each MMIO write.			
This command should not be used within a non-privilege batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation.			
This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
Default Value:		29h MI_LOAD_REGISTER_MEM	
Format:		OpCode	
22	Use Global GTT		
	Format:	Boolean	
This bit if set when executing from a non-privileged batch buffer will be treated as privilege access violation. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer or ring buffer. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.			
21	Async Mode Enable		
	Format:	Enable	
If this bit is set then the command stream will not wait for completion of this command before executing the next command. Please refer to the LOAD_INDIRECT and Predicate registers for usage of this bit.			

MI_LOAD_REGISTER_MEM

20	Reserved	Format:	MBZ
19	Add CS MMIO Start Offset	This bit controls the functionality of the Register Address field in the command.	
	Value	Name	Description
	1		Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_MEM, ADD_CS_MMIO_START_OFFSET: true, Memory Address: 0xABCD, Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a write to MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
	0	[Default]	Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.
18	Reserved	Format:	MBZ
17	MMIO Remap Enable	<p>This bit provides a mechanism in HW to remap the MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.</p> <p>A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.</p> <p>This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.</p>	
	Value	Name	Description
	1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
	0		MMIO remapping will not be applied to the MMIO address.
	Programming Notes		
	<ul style="list-style-type: none"> • SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands. • MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks. 		

MI_LOAD_REGISTER_MEM		
		<ul style="list-style-type: none"> • "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa. • When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.
	16:8	Reserved Format: _____ MBZ
	7:0	DWord Length Default Value: _____ 2h Excludes DWord (0,1) Format: _____ =n
1	31:23	Reserved Format: _____ MBZ
	22:2	Register Address Format: _____ MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	Reserved Format: _____ MBZ
2..3	63:2	Memory Address Format: _____ GraphicsAddress[63:2] This field specifies the address of the memory location where the register value specified in the DWord above will read from. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	Reserved Format: _____ MBZ

MI_LOAD_REGISTER_REG

MI_LOAD_REGISTER_REG						
Source:	CommandStreamer					
Length Bias:	2					
<p>The MI_LOAD_REGISTER_REG command reads from a source register location and writes that value to a destination register location.</p> <p>Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundary and not destination. Any other address will ensure the destination is updated prior to parsing the next command</p>						
Programming Notes						
The command temporarily halts commands that will cause cycles down the 3D pipeline.						
Destination register with mask implemented (Ex: Some registers have bits [31:16] as mask bits and bits[15:0] as data) will not get updated unless the value read from source register has the bits corresponding to the mask bits set. Note that any mask implemented register when read returns "0" for the bits corresponding to mask location. When the source and destination are mask implemented registers, destination register will not get updated with the source register contents.						
This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.						
DWord	Bit	Description				
0	31:29	Command Type				
		Default Value:	0h MI_COMMAND			
		Format:	OpCode			
	28:23	MI Command Opcode				
		Default Value:	2Ah MI_LOAD_REGISTER_REG			
		Format:	OpCode			
22:20	Reserved					
	Format:	MBZ				
19		Add CS MMIO Start Offset Destination				
		This bit controls the functionality of the Register Address Destination field in the command.				
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td> <p>Destination Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Destination Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer.</p> <p>Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: true, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030</p> <p>The above command when executed on RenderCS will result in a MMIO read from 0x1C_2130 (0x00_2000 + 0x1C_0130) and write to MMIO offset</p> </td> </tr> </tbody> </table>	Value	Name	Description	1
Value	Name	Description				
1		<p>Destination Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Destination Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer.</p> <p>Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: true, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030</p> <p>The above command when executed on RenderCS will result in a MMIO read from 0x1C_2130 (0x00_2000 + 0x1C_0130) and write to MMIO offset</p>				

MI_LOAD_REGISTER_REG											
		0x1C_2030 (0x00_2000 + 0x1C_0030) instead of read from 0x1C_0130 and write to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.									
0	[Default]	Destination Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.									
18	<p>Add CS MMIO Start Offset Source This bit controls the functionality of the Register Address Source field in the command.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td>Source Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Source Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: false, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a MMIO read from 0x1C_2130 instead of read from 0x1C_0130 and write to MMIO offset 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.</td> </tr> <tr> <td>0</td> <td>[Default]</td> <td>Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.</td> </tr> </tbody> </table>		Value	Name	Description	1		Source Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Source Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: false, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a MMIO read from 0x1C_2130 instead of read from 0x1C_0130 and write to MMIO offset 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.	0	[Default]	Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.
Value	Name	Description									
1		Source Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Source Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: false, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a MMIO read from 0x1C_2130 instead of read from 0x1C_0130 and write to MMIO offset 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.									
0	[Default]	Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.									
17	<p>MMIO Remap Enable Destination This bit provides a mechanism in HW to remap the " Destination Register" MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class. A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class. This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td>MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.</td> </tr> <tr> <td>0</td> <td></td> <td>MMIO remapping will not be applied to the MMIO address.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <ul style="list-style-type: none"> SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands. 		Value	Name	Description	1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.	0		MMIO remapping will not be applied to the MMIO address.
Value	Name	Description									
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.									
0		MMIO remapping will not be applied to the MMIO address.									

MI_LOAD_REGISTER_REG

- MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks.
- "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa.
- When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.

16 MMIO Remap Enable Source

This bit provides a mechanism in HW to remap the "Source Register" MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.

Value	Name	Description
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
0		MMIO remapping will not be applied to the MMIO address.

Programming Notes

- SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands.
- MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks.
- "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa.
- When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.

15:8 Reserved

Format:	MBZ
---------	-----

7:0 DWord Length

Default Value:	1h Excludes DWord (0,1)
Format:	=n

MI_LOAD_REGISTER_REG		
1	31:23	Reserved Format: MBZ
	22:2	Source Register Address Format: MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	Reserved Format: MBZ
2	31:23	Reserved Format: MBZ
	22:2	Destination Register Address Format: MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	Reserved Format: MBZ

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL			
Source:	BlitterCS		
Length Bias:	2		
<p>The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is inside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while outside of the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe.</p> <p>Note: The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question.</p> <p>Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	13h MI_LOAD_SCAN_LINES_EXCL
		Format:	OpCode
	22	Reserved	
		Format:	MBZ
	21:19	Display Pipe Select	
		Format:	U3
		This field selects which Display Engine (pipe) this command is targeting.	
		Value	Name
		0h	Display Pipe A
		1h	Display Pipe B
2h, 3h		Reserved	
4h		Display Pipe C	
5h	Display Pipe D		
6h, 7h	Reserved		
18:17	Reserved		
	Format:	MBZ	
5:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	

MI_LOAD_SCAN_LINES_EXCL		
		Format: =n
1	31:16	Start Scan Line Number Format: U16 This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]
	15:0	End Scan Line Number Format: U16 This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL			
Source:	RenderCS		
Length Bias:	2		
<p>The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is inside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while outside). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe. Note: The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question. Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	13h MI_LOAD_SCAN_LINES_EXCL
		Format:	OpCode
	22	Reserved	
	Format:		MBZ
	21:19	Display Pipe Select	
		Format: U3	
This field selects which Display Engine (pipe) this command is targeting.			
Value		Name	
0h		Display Pipe A	
1h		Display Pipe B	
2h		Reserved	
3h		Reserved	
4h	Display Pipe C		
5h	Display Pipe D		
18:17	Reserved		
16:6	Reserved		
	Format: MBZ		
5:0	DWord Length		
	Default Value:	0h	
	Format:	=n	

MI_LOAD_SCAN_LINES_EXCL		
1	31:29	Reserved
		Format: MBZ
	28:16	Start Scan Line Number
		Format: U13
		Range: [0, Display Buffer height in lines-1]
		This field specifies the starting scan line number of the Scan Line Window.
	15:13	Reserved
		Format: MBZ
	12:0	End Scan Line Number
		Format: U13
	This field specifies the ending scan line number of the Scan Line Window.	
	Range: [0, Display Buffer height in lines-1]	

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL			
Source:	BlitterCS		
Length Bias:	2		
<p>The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is outside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while inside of the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.</p> <p>Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	12h MI_LOAD_SCAN_LINES_INCL
		Format:	OpCode
	22	Reserved	
		Format:	MBZ
	21:19	Display Pipe Select	
		Format:	U3
		This field selects which Display Engine (pipe) this command is targeting.	
		Value	Name
0h		Display Pipe A	
1h		Display Pipe B	
2h, 3h		Reserved	
4h		Display Pipe C	
5h		Display Pipe D	
6h, 7h		Reserved	
18:17	Reserved		
16:6	Reserved		
	Format:	MBZ	
5:0	DWord Length		
	Default Value:	0h Excludes DWord (0,1)	
	Format:	=n	
1	31:16	Start Scan Line Number	

MI_LOAD_SCAN_LINES_INCL					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U16</td> </tr> <tr> <td colspan="2">This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]</td> </tr> </table>	Format:	U16	This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]	
Format:	U16				
This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]					
15:0	<p>End Scan Line Number</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U16</td> </tr> <tr> <td colspan="2">This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]</td> </tr> </table>	Format:	U16	This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]	
Format:	U16				
This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]					

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL			
Source:	RenderCS		
Length Bias:	2		
<p>The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is outside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while inside the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.</p> <p>Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	12h MI_LOAD_SCAN_LINES_INCL
		Format:	OpCode
	22	Reserved	
	Format:		MBZ
	21:19	Display Pipe Select	
		Format: U3	
This field selects which Display Engine (pipe) this command is targeting.			
Value		Name	
0h		Display Pipe A	
1h		Display Pipe B	
2h		Reserved	
3h		Reserved	
4h	Display Pipe C		
5h	Display Pipe D		
18:17	Reserved		
16:6	Reserved		
	Format: MBZ		
5:0	DWord Length		
	Default Value:	0h	
	Format:	=n	

MI_LOAD_SCAN_LINES_INCL		
1	31:29	Reserved Format: MBZ
	28:16	Start Scan Line Number Format: U13 Range: [0, Display Buffer height in lines-1] This field specifies the starting scan line number of the Scan Line window.
	15:13	Reserved Format: MBZ
	12:0	End Scan Line Number Format: U13 Range: [0, Display Buffer height in lines-1] This field specifies the ending scan line number of the Scan Line Window.

MI_MATH

MI_MATH							
Source:	VideoCS						
Length Bias:	2						
<p>The MI_MATH command allows software to send instructions to the ALU in the Command Streamer. This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported. When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.</p>							
DWord	Bit	Description					
0	31:29	Command Type					
		Default Value: 0h MI_COMMAND Format: OpCode					
	28:23	MI Command Opcode					
		Default Value: 1Ah MI_MATH Format: OpCode					
22:8	Reserved Format: MBZ						
7:0	7:0	DWord Length Format: =n					
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>[Default]</td> </tr> <tr> <td>[1,255]</td> <td></td> </tr> </tbody> </table>	Value	Name	0h	[Default]	[1,255]
	Value	Name					
	0h	[Default]					
[1,255]							
1..n	31:0	ALU INSTRUCTION					
		Format: U32					

MI_MATH

MI_MATH		
Source:	BlitterCS	
Length Bias:	2	
<p>The MI_MATH command allows software to send instructions to the ALU in the Command Streamer. This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported. When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND
		Format: OpCode
	28:23	MI Command Opcode
Default Value: 1Ah MI_MATH		
Format: OpCode		
22:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Format: =n	
	Value	Name
	0h	[Default]
	[1,255]	
1..n	31:0	ALU INSTRUCTION
		Format: U32

MI_MATH

MI_MATH						
Source:	RenderCS					
Length Bias:	2					
<p>The MI_MATH command allows SW to send instruction to ALU in Render Command Streamer. MI_MATH command is the means by which ALU can be accessed. ALU instructions form the data payload of MI_MATH command, ALU instruction is dword in size. MI_MATH Dword Length should be programmed based on the number of ALU instruction packed, max number is limited by the max Dword Length supported. When MI_MATH command is parsed by command streamer it outputs the payload dwords (ALU instructions) to the ALU. ALU takes single clock to process any given instruction. Refer to B-spec "Command Streamer (CS) ALU Programming" section in Command Streamer Programming.</p>						
DWord	Bit	Description				
0	31:29	Command Type				
		Default Value: 0h MI_COMMAND				
		Format: OpCode				
	28:23	MI Command Opcode				
Default Value: 1Ah MI_MATH						
Format: OpCode						
22:8	Reserved					
	Format: MBZ					
7:0	7:0	DWord Length				
		Format: =n				
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[0-255]</td> <td></td> </tr> </tbody> </table>		Value	Name	[0-255]	
	Value	Name				
[0-255]						
1..n	31:0	ALU INSTRUCTION				
		Format: U32				

MI_MATH

MI_MATH											
Source:	VideoEnhancementCS										
Length Bias:	2										
<p>The MI_MATH command allows software to send instructions to the ALU in the Command Streamer. This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported. When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.</p>											
DWord	Bit	Description									
0	31:29	Command Type <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>0h MI_COMMAND</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h MI_COMMAND	Format:	OpCode					
		Default Value:	0h MI_COMMAND								
		Format:	OpCode								
	28:23	MI Command Opcode <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>1Ah MI_MATH</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	1Ah MI_MATH	Format:	OpCode					
Default Value:		1Ah MI_MATH									
Format:	OpCode										
22:8	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
7:0	DWord Length <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>=n</td> </tr> <tr> <td colspan="2" style="text-align: center;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">[1,255]</td> <td></td> </tr> </tbody> </table> </td> </tr> </table>	Format:	=n	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">[1,255]</td> <td></td> </tr> </tbody> </table>		Value	Name	0h	[Default]	[1,255]	
Format:	=n										
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0h</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">[1,255]</td> <td></td> </tr> </tbody> </table>		Value	Name	0h	[Default]	[1,255]					
Value	Name										
0h	[Default]										
[1,255]											
1..n	31:0	ALU INSTRUCTION <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>U32</td> </tr> </table>	Format:	U32							
Format:	U32										

MI_NOOP

MI_NOOP			
Source:	BlitterCS		
Length Bias:	1		
<p>The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0h MI_NOOP
		Format:	OpCode
	22	Identification Number Register Write Enable	
		Format:	Enable
		<p>This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function.</p>	
		Value	Name
0h		Disable	Do not write the NOP_ID register.
1h		Enable	Write the NOP_ID register.
21:0	Identification Number		
	Format:	U22	
	<p>This field contains a 22-bit number which can be written to the MI NOPID register.</p>		

MI_NOOP

MI_NOOP			
Source:	RenderCS		
Length Bias:	1		
<p>The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).</p>			
Performance			
<p>The MI_NOOP process time is reduced to 1 clock. An example use of the improved NOOP throughput is for some multi-pass media applications where some unwanted media object commands are replaced by MI_NOOP commands without repacking the commands in a batch buffer.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0h MI_NOOP
		Format:	OpCode
	22	Identification Number Register Write Enable	
		Format:	Enable
		<p>This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified, making this command an effective "no operation" function.</p>	
		Value	Name
0h		Disable	Do not write the NOP_ID register.
1h	Enable	Write the NOP_ID register.	
21:0	Identification Number		
	Format:	U22	
<p>This field contains a 22-bit number which can be written to the MI NOPID register.</p>			

MI_NOOP

MI_NOOP			
Source:	VideoCS		
Length Bias:	1		
<p>The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	00h MI_NOOP
		Format:	OpCode
	22	Identification Number Register Write Enable	
		Format:	Enable
		<p>This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function.</p>	
		Value	Name
		1	Write the NOP_ID register.
	21:0	Identification Number	
Format:		U22	
<p>This field contains a 22-bit number which can be written to the MI NOPID register.</p>			

MI_NOOP

MI_NOOP			
Source:	VideoEnhancementCS		
Length Bias:	1		
<p>The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	00h MI_NOOP
		Format:	OpCode
	22	Identification Number Register Write Enable	
		Format:	Enable
		<p>This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function.</p>	
		Value	Name
1			Write th NOP_ID Register
0			Do not write the NOP_ID register
21:0	Identification Number		
	Format:	U22	
	<p>This field contains a 22-bit number which can be written to the MI NOPID register.</p>		

MI_PREDICATE

MI_PREDICATE			
Source:	RenderCS		
Length Bias:	1		
Programming Notes			
This command is supported by PositionCS.			
This command is supported by ComputeCS			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0Ch MI_PREDICATE
		Format:	OpCode
	22:8	Reserved	
		Format:	MBZ
	7:6	Load Operation	
		This field controls if/how the Predicate state bit is modified.	
Value		Name	Description
0h		KEEP	The Predicate state bit is unmodified.
1h		Reserved	
2h		LOAD	The Predicate state bit is loaded with the combine operation result.
3h	LOADINV	The Predicate state bit is loaded with the inverted combine operation result.	
5	Reserved		
	Format:	MBZ	
4:3	Combine Operation		
	This field controls if/how the result of the compare operation is combined with the current Predicate state bit.		
	Value	Name	Description
	0h	SET	The combine operation output the compare result unmodified.
	1h	AND	The combine operation outputs the AND of the compare result and the current Predicate state bit.
	2h	OR	The combine operation outputs the OR of the compare result and the current Predicate state bit.
3h	XOR	The combine operation outputs the XOR of the compare result and the current Predicate state bit.	

MI_PREDICATE		
2	Reserved	
	Format:	MBZ
1:0	Compare Operation This field controls how Data DWord 0 and Data DWord 1 fields are used to generate a compare operation result and possibly modify the PredicateData register.	
	Value	Name
		Description
	0h	TRUE
		The compare operation outputs TRUE. The PredicateData register is unmodified.
	1h	FALSE
		The compare operation outputs FALSE. The PredicateData register is unmodified.
	2h	SRCS_EQUAL
		(Mltemp0 - Mltemp1) is computed and loaded into the PredicateData register. The compare operation outputs (Mltemp0 == Mltemp1).
	3h	DELTAS_EQUAL
		(Mltemp0 - Mltemp1) is computed and compared to the PredicateData register. If the values are equal, the compare operation outputs TRUE, otherwise it outputs FALSE. The PredicateData register is unmodified.

MI_REPORT_HEAD

MI_REPORT_HEAD			
Source:	BlitterCS		
Length Bias:	1		
<p>The MI_REPORT_HEAD command causes the Head Pointer value of the active ring buffer to be written to a cacheable (snooped) system memory location.</p> <p>When the Execlist Enable bit is reset:</p> <p>The location written is relative to the address programmed in the Hardware Status Page Address Register.</p>			
Programming Notes			
<p>This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register). When the Execlist Disable is clear, the head pointer will be reported to the PP HW Status Page.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	07h MI_REPORT_HEAD
		Format:	OpCode
22:0	Reserved		
	Format:	MBZ	

MI_REPORT_HEAD

MI_REPORT_HEAD		
Source:	RenderCS	
Length Bias:	1	
<p>The MI_REPORT_HEAD command causes the Head Pointer value of the active ring buffer to be written to a cacheable (snooped) system memory location. When Execlist Enable is set, the head pointer will be reported to the PP HW Status Page. The location written is relative to the address programmed in the Hardware Status Page Address Register.</p>		
Programming Notes		
This command must not be executed from a Batch Buffer. (Refer to the description of the HWS_PGA register.)		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND
		Format: OpCode
	28:23	MI Command Opcode
		Default Value: 07h MI_REPORT_HEAD
		Format: OpCode
	22:0	Reserved
		Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD		
Source:	VideoCS	
Length Bias:	1	
<p>The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location. When the Per-Process Virtual Address Space and Execlist Enable bits are reset, the location written is relative to the address programmed in the Hardware Status Page Address Register. When the Execlist Enable is set, the head pointer will be reported to the PP HW Status Page.</p>		
Programming Notes		
This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND
		Format: OpCode
	28:23	MI Command Opcode
		Default Value: 07h MI_REPORT_HEAD
		Format: OpCode
	22:0	Reserved
		Format: MBZ



MI_REPORT_HEAD

MI_REPORT_HEAD			
Source:	VideoEnhancementCS		
Length Bias:	1		
<p>The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location.</p> <p>When the Per-Process Virtual Address Space and Execlist Enable bit is reset: The location written is relative to the address programmed in the Hardware Status Page Address Register. When the Execlist Enable is set, the head pointer will be reported to the PP HW Status Page.</p>			
Programming Notes			
This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	07h MI_REPORT_HEAD
		Format:	OpCode
	22:0	Reserved	
		Format:	MBZ

MI_REPORT_PERF_COUNT

MI_REPORT_PERF_COUNT			
Source:	RenderCS, ComputeCS		
Length Bias:	2		
<p>The MI_REPORT_PERF_COUNT command causes the GFX hardware to write out a snap-shot of performance counters to the address specified in this command along with constant ID field supplied and the time-stamp counter. This write is required to be treated as a cacheable write irrespective of GTT entry memory type. This command is specific to the render engine.</p>			
Programming Notes			
<p>This command can be inserted after events of interest (frequently before and after a 3DPRIMITIVE command). SW is entirely responsible for managing the ID field and addresses used by such a series of commands.</p>			
<p>GTT_SELECT must not be set to 1 (i.e. GGTT) when MI_REPORT_PERF_COUNT command is programmed in a non-privileged batch buffer. Refer to the "User Mode Privileged commands" Table in MI_BATCH_BUFFER_START command section for more details. All batch buffers in PPGTT are considered as Non-privileged.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	28h MI_REPORT_PERF_COUNT
		Format:	OpCode
	22:6	Reserved	
		Format:	MBZ
5:0	DWord Length	Default Value:	2h Excludes DWord (0,1)
		Format:	=n
	Total Length - 2		
1..2	63:6	Memory Address	
		Format:	GraphicsAddress[63:6]
		<p>This field specifies 64B aligned GFX MEM address where the chap counter values are reported. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]</p>	
	Programming Notes		
	<p>This field is ignored if "Report to OABUFFER" bit is set.</p>		
5	Reserved		
	Format:	MBZ	

MI_REPORT_PERF_COUNT					
	4	<p>Core Mode Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U1</td> </tr> </table> <p>This bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset. If Core ID 1, then the Memory is offset by the size of the data(64b).</p>	Format:	U1	
	Format:	U1			
	3:1	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ				
0	<p>Use Global GTT</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">Boolean</td> </tr> </table> <p>This field when set (i.e. bit = 1) selects the GGTT for address translation. When this bit is 0 (default value), HW should use PGTT for address translation.</p>	Format:	Boolean		
Format:	Boolean				
3	31:0	<p>Report ID</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U32</td> </tr> </table> <p>This field specifies the ID provided by SW for a given report command. It can be tracked to use different flavors of these reports based on where in command-stream they are inserted. This field is reported only when Counter Select Field is 0.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> </table> <p>If a privilege access violation occurs, the REPORT ID field in the report generated by the next legitimate MI_REPORT_PERF_COUNT will be corrupted.</p>	Format:	U32	Programming Notes
Format:	U32				
Programming Notes					

MI_RS_STORE_DATA_IMM

MI_RS_STORE_DATA_IMM			
Source:	RenderCS		
Length Bias:	2		
The MI_RS_STORE_DATA_IMM command requests a write of the DWord constant supplied in the packet to the specified Memory Address.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	2Bh MI_RS_STORE_DATA_IMM
		Format:	OpCode
			MI_RS_STORE_DATA_IMM
22	Reserved		
Format:		MBZ	
21	Reserved		
20:8	Reserved		
Format:		MBZ	
7:0	DWord Length		
	Default Value:	2h Excludes DWord (0,1)	
	Format:	=n	
1.2	63:2	Destination Address	
		Format:	GraphicsAddress[63:2]
		<p>This field specifies Bits 47:2 of the Address where the DWord will be stored. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</p> <p>When render engine is PPGTT enabled this Address is translated using PPGTT, else GGTT is used for translation.</p>	
1	Reserved		
Format:		MBZ	
0	Core Mode Enable		
<p>If this bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data.</p>			
3	31:0	Data DWord 0	
		Format:	U32



MI_RS_STORE_DATA_IMM		
		This field specifies the DWord value to be written to the targeted location.

MI_SEMAPHORE_SIGNAL

MI_SEMAPHORE_SIGNAL			
Source:	CommandStreamer		
Length Bias:	2		
Description			
<p>An engine on executing this command generates a signal (interrupt) by reporting the Producer Token Number programmed in SEMAPHORE_TOKEN register. Each engine implements its own SEMAPHORE_TOKEN register. SEMAPHORE_TOKEN register is privileged and context save/restored. Scheduler can take appropriate action on decoding the reported Producer Token Number. Typically MI_ATOMIC (non-posted) command will be used to update the memory semaphore before signaling the consumer context.</p> <p>Each engine implements SEMAPHORE_SIGNAL_PORT register for receiving semaphore signal from the scheduler (SW or FW). A write to the SEMAPHORE_SIGNAL_PORT with data as 0xFFFF_FFFF is decoded as semaphore signal received by the corresponding engine. An engine waiting on un-successful MI_SEMAPHORE_WAIT (signal mode) command will reacquire the semaphore data from memory and re-evaluate the semaphore comparison on receiving the semaphore signal. SEMAPHORE_SIGNAL_PORT register is privileged. Writing to the SEMAPHORE_SIGNAL_PORT of an idle engine (no context) does not trigger any action in HW and is of no use.</p> <p>SEMAPHORE_TOKEN, MI_SEMAPHORE_SIGNAL, SEMAPHORE_SIGNAL_PORT and MI_SEMAPHORE_WAIT together can be used to create semaphores between producer context and consumer context.</p> <p>MI_SEMAPHORE_SIGNAL command from a producer context can be used to signal a consumer context waiting on MI_SEMAPHORE_WAIT (signal mode) command through scheduler (SW or FW).</p> <ul style="list-style-type: none"> Typically MI_ATOMIC (non-posted) command will be used to update the memory semaphore by the producer context before signaling the consumer context. Scheduler on receiving the signal will process the Producer Token Number and if required will signal the consumer context running on an engine by writing 0xFFFF_FFFF to the corresponding engines SEMAPHORE_SIGNAL_PORT. A consumer context will wait on MI_SEMAPHORE_WAIT (signal mode) command until the semaphore comparison is successful. An engine waiting on un-successful MI_SEMAPHORE_WAIT (signal mode) command will reacquire the semaphore data from memory and re-evaluate the semaphore comparison on receiving the semaphore signal. MI_SEMAPHORE_WAIT command has Wait Token Number as inline data programmed by the SW. Context switched out an un-successful MI_SEMAPHORE_WAIT command will report Wait Token Number as Wait Detail field in the CSB structure. 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	1Bh MI_SEMAPHORE_SIGNAL
		Format:	OpCode
22	Reserved		

MI_SEMAPHORE_SIGNAL

		Format:	MBZ
21	Post-Sync Operation		
	Source:		RenderCS
	Value	Name	Description
	0h	No Post Sync Operation	Command is executed as usual.
	1h	Post Sync Operation	MI_SEMAPHORE_SIGNAL command is executed as a pipelined PIPE_CONTROL flush command with Semaphore Signal as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any outstanding flushes issued prior to this command.
Programming Notes			
<p>Any desired pipeline flush operation can be achieved by programming PIPE_CONTROL command prior to this command.</p> <p>When this bit is set Command Streamer sends a flush down the pipe and the atomic operation is saved as post sync operation. Command streamer goes on executing the following commands. Atomic operation saved as post sync operation is executed at some point later on completion of corresponding flush issued.</p> <p>When this bit is set atomic semaphore signal operation will be out of order with rest of the MI commands programmed in the ring buffer or batch buffer, it will be in order with respect to the post sync operations resulting due to PIPE_CONTROL command.</p> <p>This bit must not be set when executed by ComputeCS.</p>			
20:8	Reserved		
	Format:		MBZ
7:0	DWord Length		
	Default Value:		0h Excludes DWord (0,1)
	Format:		=n
	Total Length - 2		
1	31:0	Reserved	
		Format:	
		MBZ	

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT	
Source:	CommandStreamer
Length Bias:	2
Description	
<p>This command supports memory based Semaphore WAIT. Memory based semaphores will be used for synchronization between the Producer and the Consumer contexts. Producer and Consumer Contexts could be running on different engines or on the same engine inside GT. Producer Context implements a Signal and Consumer context implements a Wait.</p> <p>Command Streamer on parsing this command fetches data from the Semaphore Address mentioned in this command and compares it with the inline Semaphore Data Dword.</p> <ul style="list-style-type: none"> • If comparison passes, the command streamer moves to the next command. • If comparison fails Command streamer switches out the context. Context switch can be inhibited by setting "Inhibit Synchronous Context Switch" in CTXT_SR_CTL register. • If "Inhibit Synchronous context Switch" is enabled and comparison fails, Command Streamer evaluates the Compare Operation based on the Wait Mode until the compare operation is true or Wait is canceled by SW. • CS generates semaphore wait interrupt to the scheduler when MI_SEMAPHORE_WAIT command is un-successful and when "Inhibit Synchronous Context Switch" is set. Scheduler can use this interrupt to preempt the context waiting on semaphore wait. <p>MI_SEMAPHORE_WAIT command also supports register based Semaphore WAIT. Command Streamer on parsing this command fetches data from the MMIO offset mentioned in this command and compares it with the inline Semaphore Data Dword. This functionality is supported when Register Poll bit is set in the command header. In register poll mode of operation Wait Mode supported is always Poll mode and no Signal mode is supported.</p> <ul style="list-style-type: none"> • If comparison passes, the command streamer moves to the next command. • Unlike in Memory based semaphore, there is no context switch on an un-successful semaphore wait in Register Poll mode, however preemption is supported on unsuccessful semaphore wait in Register Poll mode. Semaphore wait interrupt is not generated by default on wait un-successful in Register Poll mode. • Also unlike in Memory based semaphore, generation of an interrupt for a semaphore wait in "Register Poll" mode is not dependent on the value of bit "Inhibit Synchronous Context Switch" in register "CTXT_SR_CTL" • Register Poll mode of Semaphore Wait command operation is non-privileged and will be supported from PPGTT batch buffers. • HW will trigger Render DOP CG on semaphore wait unsuccessful by default and can be disabled if not desired by programming Register Poll Mode Semaphore Wait Event IDLE message Disable bit in INSTPM register. Note that Render DOP CG will not be triggered on register semaphore wait un-successful from INDIRECT_CTX pointer or BB_PER_CTX_PTR buffers. 	
Programming Notes	

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT command must not be used in the middle of a tile pass on the posh pipe.

DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	1Ch MI_SEMAPHORE_WAIT
		Format:	OpCode
	22	Memory Type	
		This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit <i>must</i> be 1 if the Per Process GTT Enable bit is clear.	
		Value	Name
		0h	Per Process Graphics Address
		1h	Global Graphics Address
	21:19	Reserved	
Format:		MBZ	
18	Reserved		
	Format:	MBZ	
17	Reserved		
16	Register Poll Mode		
	This field control the semaphore wait behavior of polling from memory vs MMIO register.		
	Value	Name	
	1h	Register Poll [Default]	In this mode HW periodically reads the semaphore data from MMIO register instead of memory for comparison until the condition is satisfied. Periodicity will be mentioned in a SEMA_WAIT_POLL register. When operating in register poll mode, DW2 Semaphore Address (bits 22:2) carries the register MMIO offset to be polled. In register poll mode Memory Type field of this command are ignored by HW.
	0h	Memory Poll	In this mode HW will functional as in regular mode and checks for semaphore data in memory.
Programming Notes			
In register poll mode of operation of MI_SEMAPHORE_WAIT command, context switch is not supported on un-successful wait. Wait Mode must be always set to Polling Mode when			

MI_SEMAPHORE_WAIT

	<p>Register Poll Mode is enabled. Preemption is supported on unsuccessful semaphore wait in Register Poll mode if operation.</p>																												
15	<p>Wait Mode This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>1h</td> <td>Polling Mode</td> <td>In this mode HW periodically reads the semaphore data from memory for comparison until it is context switched out. Periodicity will be mentioned in a SEMA_WAIT_POLL register.</td> </tr> <tr> <td>0h</td> <td>Signal Mode</td> <td>[] In this mode HW will reacquire the semaphore data from memory for evaluating semaphore wait condition on receiving SIGNAL. Scheduler or SW can generate a SIGNAL to an engine by writing a value 0xFFFF_FFFF to the engines corresponding SEMAPHORE_SIGNAL_PORT register.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Wait Mode must be always set to Polling Mode when Register Poll Mode is enabled.</p>		Value	Name	Description	1h	Polling Mode	In this mode HW periodically reads the semaphore data from memory for comparison until it is context switched out. Periodicity will be mentioned in a SEMA_WAIT_POLL register.	0h	Signal Mode	[] In this mode HW will reacquire the semaphore data from memory for evaluating semaphore wait condition on receiving SIGNAL. Scheduler or SW can generate a SIGNAL to an engine by writing a value 0xFFFF_FFFF to the engines corresponding SEMAPHORE_SIGNAL_PORT register.																		
Value	Name	Description																											
1h	Polling Mode	In this mode HW periodically reads the semaphore data from memory for comparison until it is context switched out. Periodicity will be mentioned in a SEMA_WAIT_POLL register.																											
0h	Signal Mode	[] In this mode HW will reacquire the semaphore data from memory for evaluating semaphore wait condition on receiving SIGNAL. Scheduler or SW can generate a SIGNAL to an engine by writing a value 0xFFFF_FFFF to the engines corresponding SEMAPHORE_SIGNAL_PORT register.																											
14:12	<p>Compare Operation This field specifies the operation that will be executed to create the result that will either allow the context to continue or wait.</p> <p>SAD = Semaphore Address Data SDD = Semaphore Data Dword</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 45%;">Name</th> <th style="width: 45%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>SAD_GREATER_THAN_SDD</td> <td>If Indirect fetched data is greater than inline data then continue.</td> </tr> <tr> <td>1h</td> <td>SAD_GREATER_THAN_OR_EQUAL_SDD</td> <td>If Indirect fetched data is greater than or equal to inline data then continue.</td> </tr> <tr> <td>2h</td> <td>SAD_LESS_THAN_SDD</td> <td>If Indirect fetched data is less than inline data then continue.</td> </tr> <tr> <td>3h</td> <td>SAD_LESS_THAN_OR_EQUAL_SDD</td> <td>If Indirect fetched data is less than or equal to inline data then continue.</td> </tr> <tr> <td>4h</td> <td>SAD_EQUAL_SDD</td> <td>If Indirect fetched data is equal to inline data then continue.</td> </tr> <tr> <td>5h</td> <td>SAD_NOT_EQUAL_SDD</td> <td>If Indirect fetched data is not equal to inline data then continue.</td> </tr> <tr> <td>6h</td> <td>Reserved</td> <td></td> </tr> <tr> <td>7h</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>		Value	Name	Description	0h	SAD_GREATER_THAN_SDD	If Indirect fetched data is greater than inline data then continue.	1h	SAD_GREATER_THAN_OR_EQUAL_SDD	If Indirect fetched data is greater than or equal to inline data then continue.	2h	SAD_LESS_THAN_SDD	If Indirect fetched data is less than inline data then continue.	3h	SAD_LESS_THAN_OR_EQUAL_SDD	If Indirect fetched data is less than or equal to inline data then continue.	4h	SAD_EQUAL_SDD	If Indirect fetched data is equal to inline data then continue.	5h	SAD_NOT_EQUAL_SDD	If Indirect fetched data is not equal to inline data then continue.	6h	Reserved		7h	Reserved	
Value	Name	Description																											
0h	SAD_GREATER_THAN_SDD	If Indirect fetched data is greater than inline data then continue.																											
1h	SAD_GREATER_THAN_OR_EQUAL_SDD	If Indirect fetched data is greater than or equal to inline data then continue.																											
2h	SAD_LESS_THAN_SDD	If Indirect fetched data is less than inline data then continue.																											
3h	SAD_LESS_THAN_OR_EQUAL_SDD	If Indirect fetched data is less than or equal to inline data then continue.																											
4h	SAD_EQUAL_SDD	If Indirect fetched data is equal to inline data then continue.																											
5h	SAD_NOT_EQUAL_SDD	If Indirect fetched data is not equal to inline data then continue.																											
6h	Reserved																												
7h	Reserved																												
11:10	<p>Reserved</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>		Format:	MBZ																									
Format:	MBZ																												

MI_SEMAPHORE_WAIT			
	9:8	Reserved Format: _____ MBZ	
	7:0	DWord Length Format: _____ =n	
		Value	Name
		3h	Excludes DWord (0,1) [Default]
1	31:0	Semaphore Data Dword Format: _____ U32 This Data dword is supplied by software to control execution of the command buffer. This value is used as part of the comparison to result in waiting or continuing in the command parser if enabled.	
2..3	63:2	Semaphore Address Format: _____ VIRTUAL_ADDR[63:2] Register Poll Mode: In Register Poll mode of operation, Bits 22:2 (Bits 63:23 are reserved MBZ, HW enforced) specify the MMIO offset of the register for the semaphore. Non Register Poll Mode: This field is the Graphics Memory Address of the 32-bit value for the semaphore. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form.	
	1:0	Reserved Format: _____ MBZ	
4	31:22	Reserved Format: _____ MBZ	
	21:10	Reserved Format: _____ MBZ	
	9:5	Wait Token Number When context is switched out due to Semaphore wait, WaitTokenNumber is reported as Wait Detail in the CSB structure.	
	4:2	Reserved Format: _____ MBZ	
	1:0	Reserved Format: _____ MBZ	

MI_SET_CONTEXT

MI_SET_CONTEXT			
Source:	RenderCS		
Length Bias:	2		
<p>The MI_SET_CONTEXT command is used to specify the logical context associated with the hardware context. A logical context is an area in memory used to store hardware context information, and the context is referenced via a 2KB-aligned pointer. If the (new) logical context is different (i.e., at a different memory address), the device saves the current HW context values to the current logical context address, and then restores (loads) the new logical context by reading the context from the new address and loading it into the hardware context state. If the logical context address specified in this command matches the current logical context address, this command is effectively treated as a NOOP. Specific to the Render command stream only. This command also includes some controls over the context save/restore process. The Force Restore bit can be used to refresh the on-chip device state from the same memory address if the indirect state buffers have been modified. The Restore Inhibit bit can be used to prevent the new context from being loaded at all. This must be used to prevent an uninitialized context from being loaded. Once software has initialized a context (by setting all state variables to initial values via commands), the context can then be stored and restored normally. When switching from a generic media context to a 3D context, the generic media state must be cleared via the Generic Media State Clear bit 16 in PIPE_CONTROL (or bit 4 in MI_FLUSH) before saving 3D context. MI_SET_CONTEXT commands are permitted only within a ring buffer (not within a batch buffer). All context is saved and restored from a GGTT space. This command does not initiate any interrupt due to context switch of any kind and does not support any workaround batch buffer or indirect context offset feature.</p>			
Programming Notes			
For ring buffer mode, the first 128B(2 cache lines) of the context image are saved as zeros.			
In execution list mode, this command must be preceded with a MI_ARB_ON_OFF command to disable arbitration and followed by a MI_ARB_ON_OFF command to enable arbitration. The first 320 bytes(5 cache lines) of the context image will be saved as zeros.			
Arbitration Mode must be set to not allow lite restore prior to this command being executed. This bit is a field in the MI_ARB_ON_OFF command when in Execution list Mode.			
This command needs to be always followed by a single MI_NOOP instruction to workaround a silicon issue.			
MI_ARB_ON_OFF with 'Arbitration Enable Reset' set should be programmed before an MI_SET_CONTEXT command. MI_ARB_ON_OFF with 'Arbitration Enable' set should be programmed after an MI_SET_CONTEXT command.			
MI_SET_CONTEXT command must not be programmed for a POSH enabled context.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	18h MI_SET_CONTEXT

MI_SET_CONTEXT		
		Format: OpCode
	22:8	Reserved Format: MBZ
	7:0	DWord Length Default Value: 0h Format: =n
1	31:12	Logical Context Address Format: GraphicsAddress[31:12]
		This field contains the 4KB-aligned graphics memory address of the Logical Context that is to be loaded into the hardware context. If this address is equal to the CCID register associated with the current ring, no load will occur. Prior to loading this new context, the device will save the existing context as required. After the context switch operation completes, this address will be loaded into the associated CCID register.
		This field needs to be 4KB aligned virtual address.
	11:9	Reserved Format: MBZ
	8	Reserved, Must be 1 Format: MBO
	7:5	Reserved Format: MBZ
	4	Core Mode Enable Format: Enable If set the Context Image will be offset based off the Core ID: If Core ID 0, no offset If Core ID 1, 36KB Offset
	3	Resource Streamer State Save Enable Format: Enable If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is saved as part of switching away from this logical context. This bit will be stored in the associated CCID register to control the context save operation when switching away from this context (as part of a subsequent MI_SET_CONTEXT command).
	2	Resource Streamer State Restore Enable Format: Enable If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is loaded (or restored) as part of switching to this logical context. This bit affects the switch (if required) to the context specified in Logical Context Address. This bit will also be stored in the associated CCID register to control a subsequent context save operation

MI_SET_CONTEXT	
	when switching to this context (as part of a subsequent ring buffer switch).
1	<p>Force Restore</p> <p>When switching to this logical context a comparison between Logical Context Address and the contents of the CCID register is performed. Normally, matching addresses prevent a context restore from occurring; however, when this bit is set a context restore is forced to occur. This bit cannot be set with Restore Inhibit. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.</p>
0	<p>Restore Inhibit</p> <p>If set, the restore of the HW context from the logical context specified by Logical Context Address is inhibited (i.e., the existing HW context values are maintained). This bit must be used to prevent the loading of an uninitialized logical context. If clear, the context switch proceeds normally. This bit cannot be set with Force Restore. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.</p>



MI_SET_PREDICATE

MI_SET_PREDICATE			
Source:	CommandStreamer		
Length Bias:	1		
Description			
<p>This command provides a mechanism to NOOP a section of commands programmed in the command buffer. This command on execution evaluates the condition based on the Predicate Enable field and sets the predicate status accordingly in HW. On predicate status set, HW NOOPS the commands subsequently parsed until the predicate status is re-evaluated and reset on executing MI_SET_PREDICATE. MI_SET_PREDICATE is the only command executed by HW when parsed during predicate status set. Resource Streamer doesn't take any action on parsing this command.</p>			
Programming Notes			
<ul style="list-style-type: none"> MI_SET_PREDICATE predication scope must be confined within a Batch Buffer to set of commands. MI_SET_PREDICATE with Predicate Enable Must always have a corresponding MI_SET_PREDICATE with Predicate Disable within the same Batch Buffer. 			
<p>Only the following command(s) can be programmed between the MI_SET_PREDICATE command enabled for predication: 3DSTATE_URB_VS 3DSTATE_URB_HS 3DSTATE_URB_DS 3DSTATE_URB_GS 3DSTATE_PUSH_CONSTANT_ALLOC_VS 3DSTATE_PUSH_CONSTANT_ALLOC_HS 3DSTATE_PUSH_CONSTANT_ALLOC_DS 3DSTATE_PUSH_CONSTANT_ALLOC_GS 3DSTATE_PUSH_CONSTANT_ALLOC_PS MI_LOAD_REGISTER_IMM MEDIA_VFE_STATE MEDIA_OBJECT MEDIA_OBJECT_WALKER MEDIA_INTERFACE_DESCRIPTOR_LOAD 3DSTATE_WM_HZ_OP MI_STORE_DATA_IMM</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	01h MI_SET_PREDICATE
		Format:	OpCode
	22:6	Reserved	
		Format:	MBZ
	5:4	Reserved	
		Format:	MBZ
	3:0	Predicate Enable	
		Source:	RenderCS, PositionCS, ComputeCS

MI_SET_PREDICATE

Value	Name	Description
0h	NOOP Never	Predication is Disabled and CS will process commands as usual.
1h	NOOP on Result2 clear	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is clear.
2h	NOOP on Result2 set	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is set.
3h	NOOP on Result clear	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT is clear.
4h	NOOP on Result set	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT is set.
5h, 6h, 7h, 8h, 9h, Ah	Reserved	
Bh, Ch, Dh, Eh	Reserved	
Fh	NOOP Always	Following Commands will be NOOPED by RCS unconditionally.

3:0	Predicate Enable	
Source:	BlitterCS, VideoCS, VideoEnhancementCS	

Value	Name	Description
0h	Predicate Disable	Predication is Disabled and CS will process commands as usual.
1h	NOOP on Result2 Clear	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is clear.
2h	NOOP on Result2 Set	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is set.
3h,4h,5h, 6h, 7h, 8h, 9h, Ah, Bh, Ch, Dh, Eh	Reserved	
Fh	NOOP Always	Following Commands will be NOOPED by CS unconditionally.

MI_STORE_DATA_IMM

MI_STORE_DATA_IMM			
Source:	CommandStreamer		
Length Bias:	2		
<p>The MI_STORE_DATA_IMM command requests a write of the QWord constant supplied in the packet to the specified Memory Address. As the write targets a System Memory Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).</p> <p>This command supports writing to multiple consecutive dwords or qwords memory locations from the starting address.</p>			
Programming Notes			
<ul style="list-style-type: none"> This command should not be used within a "non-privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space. This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers). This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations. 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
Default Value:		20h MI_STORE_DATA_IMM	
22		Use Global GTT	
		Format:	Boolean
21		Store Qword	
		Format:	Boolean
<p>If set, this command generates Qword writes to memory, two "Data Dword" are paired to form a Qword. Number of qwords generated depends upon the number of "Data Dword" programmed</p>			

MI_STORE_DATA_IMM									
	<p>in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x/2" qword writes to memory. If reset this command generates Dwords writes to memory. Number of dwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x" dword writes to memory.</p>								
20:13	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
12	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
11	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
10	<p>Reserved</p>								
9:0	<p>DWord Length</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">=n</td> </tr> </table> <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 70%;">Name</th> </tr> </thead> <tbody> <tr> <td>2h</td> <td>Store Dword [Default]</td> </tr> <tr> <td>3h</td> <td>Store Qword</td> </tr> </tbody> </table> <p style="text-align: center; margin-top: 10px;">Programming Notes</p> <p>DWord Length programmed must not exceed 0x3FE.</p> <p>If RS is enabled in the batch buffer, then the value of this field must not exceed 0x3F.</p>	Format:	=n	Value	Name	2h	Store Dword [Default]	3h	Store Qword
Format:	=n								
Value	Name								
2h	Store Dword [Default]								
3h	Store Qword								
1..2	<p>63:2 Address</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Format:</td> <td style="width: 60%;">VIRTUAL_ADDR[63:2]</td> </tr> </table> <p>GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost bits are ignored and MBZ. This field specifies Bits 47:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.</p>	Format:	VIRTUAL_ADDR[63:2]						
Format:	VIRTUAL_ADDR[63:2]								
	<p>1 Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
	<p>0 Core Mode Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U1</td> </tr> </table> <p>This bit is set then the address will be offset by the Core ID: If Core ID 0, then there is no offset. If Core ID 1, then the Memory is offset by the size of the data (32b or 64b based off number of DW length).</p>	Format:	U1						
Format:	U1								
3	<p>31:0 Data DWord 0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">U32</td> </tr> </table>	Format:	U32						
Format:	U32								

MI_STORE_DATA_IMM				
		This field specifies the DWord value to be written to the targeted location. For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).		
4	31:0	<p>Data DWord 1</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>U32</td> </tr> </table> <p>This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).</p>	Format:	U32
Format:	U32			

MI_STORE_DATA_INDEX

		MI_STORE_DATA_INDEX	
Source:	CommandStreamer		
Length Bias:	2		
<p>The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).</p>			
Programming Notes			
<ul style="list-style-type: none"> Use of this command with an invalid or uninitialized value in the Hardware Status Page Address Register is UNDEFINED. This command can be used for general software synchronization through variables in cacheable memory(i.e., where software does not need to poll uncached memory or device registers). This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations. 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	21h MI_STORE_DATA_INDEX
	22	Reserved	
		Format:	MBZ
21	Use Per-Process Hardware Status Page If this bit is set, this command will index into the per-process hardware status page at offset 0K from the LRCA. If clear, the Global Hardware Status Page will be indexed.		
20:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	1h	
	Format:	=n	
1	31:12	Reserved	
		Format:	MBZ

MI_STORE_DATA_INDEX				
	11:2	Offset		
		Format: U10 zero-based DWord offset into the HW status page.		
		This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage - targeting these reserved locations via this command is UNDEFINED. This address must be 8B aligned for a store QW command.		
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[16, 1023]</td> <td></td> </tr> </tbody> </table>	Value	Name
Value	Name			
[16, 1023]				
1:0	Reserved			
	Format: MBZ			
2	31:0	Data DWord 0		
		Format: U32		
		This field specifies the DWord value to be written to the targeted location. For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).		
3	31:0	Data DWord 1		
		Format: U32		
		This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).		

MI_STORE_REGISTER_MEM

MI_STORE_REGISTER_MEM			
Source:	CommandStreamer		
Length Bias:	2		
<p>The MI_STORE_REGISTER_MEM command requests a register read from a specified memory mapped register location in the device and store of that DWord to memory. The register address is specified along with the command to perform the read.</p>			
Programming Notes			
<ul style="list-style-type: none"> The command temporarily halts command execution. The memory address for the write is snooped on the host bus. This command should not be used from within a "non-privilege" batch buffer to access global virtual space. doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or "privilege" batch buffers to access global virtual space. This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers. 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
Default Value:		24h MI_STORE_REGISTER_MEM	
Format:		OpCode	
22	Use Global GTT		
	Format:	Boolean	
		<p>It is allowed for this bit to be set when executing this command from a privileged (secure) batch or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</p>	
21	Predicate Enable		
	Source:	RenderCS, PositionCS, ComputeCS	
		<p>If set, this command is executed (or not) depending on the current value of the MI_PREDICATE internal state bit in MMIO register MI_PREDICATE_RESULT[0]. This command is ignored only if PredicateEnable is set and value in the MMIO register MI_PREDICATE_RESULT[0] is 0.</p>	

MI_STORE_REGISTER_MEM

20	Reserved		MBZ
	Format:		
19	Add CS MMIO Start Offset	This bit controls the functionality of the Register Address field in the command.	
	Value	Name	Description
	1		Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_STORE_REGISTER_MEM, ADD_CS_MMIO_START_OFFSET: true, Memory Address:0xABCD, Register Address: 0x1C_0030 The above command when executed on RenderCS will result in updating the memory address with the content of the MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
	0	[Default]	Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.
18	Reserved		MBZ
	Format:		
17	MMIO Remap Enable	<p>This bit provides a mechanism in HW to remap the MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.</p> <p>A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.</p> <p>This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.</p>	
	Value	Name	Description
	1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
	0		MMIO remapping will not be applied to the MMIO address.
Programming Notes			
<ul style="list-style-type: none"> SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands. 			

MI_STORE_REGISTER_MEM															
	<ul style="list-style-type: none"> MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks. "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa. When remapping is not found in the remap table, HW will use the MMIO address directly without any modification. 														
	<table border="1"> <tr> <td>16:8</td> <td>Reserved</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	16:8	Reserved	Format:	MBZ										
16:8	Reserved														
Format:	MBZ														
	<table border="1"> <tr> <td>7:0</td> <td>DWord Length</td> </tr> <tr> <td>Default Value:</td> <td>2h Excludes DWord (0,1)</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table>	7:0	DWord Length	Default Value:	2h Excludes DWord (0,1)	Format:	=n								
7:0	DWord Length														
Default Value:	2h Excludes DWord (0,1)														
Format:	=n														
1	<table border="1"> <tr> <td>31:23</td> <td>Reserved</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> <tr> <td>22:2</td> <td> <table border="1"> <tr> <td>Register Address</td> </tr> <tr> <td>Format: MMIOAddress[22:2]</td> </tr> </table> <p>This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.</p> <table border="1"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td> <ul style="list-style-type: none"> Storing a VGA register is not permitted and will store an UNDEFINED value. The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified. </td> </tr> </table> </td> </tr> <tr> <td>1:0</td> <td> <table border="1"> <tr> <td>Reserved</td> </tr> <tr> <td>Format: MBZ</td> </tr> </table> </td> </tr> </table>	31:23	Reserved	Format:	MBZ	22:2	<table border="1"> <tr> <td>Register Address</td> </tr> <tr> <td>Format: MMIOAddress[22:2]</td> </tr> </table> <p>This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.</p> <table border="1"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td> <ul style="list-style-type: none"> Storing a VGA register is not permitted and will store an UNDEFINED value. The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified. </td> </tr> </table>	Register Address	Format: MMIOAddress[22:2]	Programming Notes	<ul style="list-style-type: none"> Storing a VGA register is not permitted and will store an UNDEFINED value. The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified. 	1:0	<table border="1"> <tr> <td>Reserved</td> </tr> <tr> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ
31:23	Reserved														
Format:	MBZ														
22:2	<table border="1"> <tr> <td>Register Address</td> </tr> <tr> <td>Format: MMIOAddress[22:2]</td> </tr> </table> <p>This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.</p> <table border="1"> <tr> <td style="text-align: center;">Programming Notes</td> </tr> <tr> <td> <ul style="list-style-type: none"> Storing a VGA register is not permitted and will store an UNDEFINED value. The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified. </td> </tr> </table>	Register Address	Format: MMIOAddress[22:2]	Programming Notes	<ul style="list-style-type: none"> Storing a VGA register is not permitted and will store an UNDEFINED value. The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified. 										
Register Address															
Format: MMIOAddress[22:2]															
Programming Notes															
<ul style="list-style-type: none"> Storing a VGA register is not permitted and will store an UNDEFINED value. The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified. 															
1:0	<table border="1"> <tr> <td>Reserved</td> </tr> <tr> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ												
Reserved															
Format: MBZ															
2..3	<table border="1"> <tr> <td>63:2</td> <td> <table border="1"> <tr> <td>Memory Address</td> </tr> <tr> <td>Format: GraphicsAddress[63:2]</td> </tr> </table> <p>This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</p> </td> </tr> <tr> <td>1:0</td> <td> <table border="1"> <tr> <td>Reserved</td> </tr> <tr> <td>Format: MBZ</td> </tr> </table> </td> </tr> </table>	63:2	<table border="1"> <tr> <td>Memory Address</td> </tr> <tr> <td>Format: GraphicsAddress[63:2]</td> </tr> </table> <p>This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</p>	Memory Address	Format: GraphicsAddress[63:2]	1:0	<table border="1"> <tr> <td>Reserved</td> </tr> <tr> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ						
63:2	<table border="1"> <tr> <td>Memory Address</td> </tr> <tr> <td>Format: GraphicsAddress[63:2]</td> </tr> </table> <p>This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</p>	Memory Address	Format: GraphicsAddress[63:2]												
Memory Address															
Format: GraphicsAddress[63:2]															
1:0	<table border="1"> <tr> <td>Reserved</td> </tr> <tr> <td>Format: MBZ</td> </tr> </table>	Reserved	Format: MBZ												
Reserved															
Format: MBZ															



MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH			
Source:	BlitterCS		
Length Bias:	1		
Blocks PM Flush Requests.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0Bh MI_SUSPEND_FLUSH
		Format:	OpCode
	22:1	Reserved	
		Format:	MBZ
	0	Suspend Flush	
		Format:	Enable
This field suspends flush due to a PM flush request.			

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH			
Source:	RenderCS		
Length Bias:	1		
Blocks PM Flush Requests.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0Bh MI_SUSPEND_FLUSH
		Format:	OpCode
	22:1	Reserved	
		Format:	MBZ
	0	Suspend Flush	
		Format:	Enable
This field suspends flush due to a PM flush request.			



MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH			
Source:	VideoCS		
Length Bias:	1		
Blocks PM Flush Requests.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0Bh MI_SUSPEND_FLUSH
		Format:	OpCode
	22:1	Reserved	
		Format:	MBZ
	0	Suspend Flush	
		Format:	Enable
	This field suspends flush due to a PM flush request.		

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH			
Source:	VideoEnhancementCS		
Length Bias:	1		
Blocks PM Flush Requests.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0Bh MI_SUSPEND_FLUSH
		Format:	OpCode
	22:1	Reserved	
		Format:	MBZ
	0	Suspend Flush	
		Format:	Enable
	This field suspends flush due to a PM flush request.		

MI_TOPOLOGY_FILTER

MI_TOPOLOGY_FILTER			
Source:	RenderCS		
Length Bias:	1		
<p>This command is used to specify a specific 3DPrimType value, where the CS will ignore all 3DPRIMITIVE commands that do not have a matching 3DPrimType. This primitive culling is optional (turned off by using this command with a Topology Filter Value of 0). This command is specific to the Render command stream only.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	0Dh MI_TOPOLOGY_FILTER
		Format:	OpCode
	22:6	Reserved	
		Format:	MBZ
	5:0	Topology Filter Value	
		Format:	3D_Prim_Topo_Type
	<p>When non-zero, the CS will discard all 3DPRIMITIVE commands which do not match the specified 3DPrimTopologyType. When zero, no filtering is performed (normal operation).</p>		

MI_UPDATE_GTT

MI_UPDATE_GTT			
Source:	BSpec		
Length Bias:	2		
<p>The MI_UPDATE_GTT command is used to update GTT page table entries in a coherent manner and at a predictable place in the command flow.</p> <p>A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries.</p> <p>PIPE_CONTROL flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering).</p> <p>MI_UPDATE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer.</p> <p>PPGTT updates cannot be done via MI_UPDATE_GTT, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.</p>			
<p>The MI_UPDATE_GTT command is used to update GGTT page table entries in a coherent manner and at a predictable place in the command flow. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries. MI_FLUSH_DWORD flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering). MI_UPDATE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer. PPGTT updates cannot be done via MI_UPDATE_GTT, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	23h MI_UPDATE_GTT
		Format:	OpCode
	22:10	Reserved	
		Format:	MBZ

MI_UPDATE_GTT			
	9:0	DWord Length	
		Format: =n	
		n = 2b (where b = # of Entry Data included)	
		Value	Name
		[2,1022]	
	2	[Default]	
1	31:12	Entry Address	
	Format: GraphicsAddress[31:12] This field holds the QW offset of the first table entry to be modified in GGTT.		
	11:0	Reserved	
Format: MBZ			
2..n	63:0	Entry Data	
Format: U64			
This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.			

MI_USER_INTERRUPT

MI_USER_INTERRUPT			
Source:	BlitterCS		
Length Bias:	1		
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	02h MI_USER_INTERRUPT
		Format:	OpCode
	22:0	Reserved	
		Format:	MBZ



MI_USER_INTERRUPT

MI_USER_INTERRUPT			
Source:	RenderCS		
Length Bias:	1		
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	02h MI_USER_INTERRUPT
		Format:	OpCode
	22:0	Reserved	
		Format:	MBZ

MI_USER_INTERRUPT

MI_USER_INTERRUPT			
Source:	VideoCS		
Length Bias:	1		
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	02h MI_USER_INTERRUPT
		Format:	OpCode
	22:0	Reserved	
		Format:	MBZ



MI_USER_INTERRUPT

MI_USER_INTERRUPT			
Source:	VideoEnhancementCS		
Length Bias:	1		
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	0h MI_COMMAND
		Format:	OpCode
	28:23	MI Command Opcode	
		Default Value:	02h MI_USER_INTERRUPT
		Format:	OpCode
	22:0	Reserved	
		Format:	MBZ

MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT		
Source:	RenderCS, BlitterCS	
Length Bias:	1	
<p>The MI_WAIT_FOR_EVENT command is used to pause command stream processing of an engine (render or blitter) until a specific display event occurs (V-Blank) or a specific condition (Flip Pending, Scanline Pending) exists.</p> <ul style="list-style-type: none"> • Display engine can be configured to generate periodic V-Blank event to an engine. • An engine on executing MI_LOAD_SCANLINE_INCL/EXCL command for a display pipe, expects a corresponding scanline event response from display engine. Engine tracks the pending scanline response for each of the display pipe separately. • An engine on executing MI_DISPLAY_FLIP command for a display plane, expects a corresponding flip done event response from display engine. Engine tracks the pending flip done response (flip pending) for each of the display plane separately. Display flip could be of type Sync Flip or Async Flip, hence engine also tracks the type of flip (Sync or Async) along with the pending flip done response for a given display plane. <p>Only one event or condition can be specified in the command -- specifying multiple events or conditions is UNDEFINED. The command parser will halt until the event occurs or condition exists on parsing this command. Note that if a specified condition (Pending Flip Done response or Pending Scanline response) does not exist at the time the parser executes this command, the parser proceeds, treating this command as a no-operation (Ex: Command is no-operation when parsed with Display Plane Flip Pending Wait Enable set to Display Plane-1 when there is no outstanding flip done response for Display Plane-1 in the engine).</p> <p>Execution List Mode of Scheduling:</p> <p>An engine on evaluating unsuccessful MI_WAIT_FOR_EVENT(results in pausing command stream) triggers synchronous context switch stating the switch reason in Context Status Buffer. With exception of not triggering synchronous context switch on unsuccessful MI_WAIT_FOR_EVENT due to Flip Pending on an Async flip. Note that synchronous context switch can be inhibited through programming Inhibit Synchronous Context Switch bit in CTXT_SR_CTL register or by disabling arbitration through MI_ARB_ON_OFF command around MI_WAIT_FOR_EVENT. When synchronous context switch is inhibited and the engine is waiting on an unsuccessful MI_WAIT_FOR_EVENT, a submission of new execlist will trigger preemption process switching out the context. With exception of not triggering preemption on an unsuccessful MI_WAIT_FOR_EVENT due to Flip Pending on an Async flip.</p> <p>Engine will always re-evaluate the wait condition for context switched out due to unsuccessful MI_WAIT_FOR_EVENT on resubmission of the context.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND
	Format: OpCode	
	28:23	MI Command Opcode
Default Value: 03h MI_WAIT_FOR_EVENT		
Format: OpCode		

MI_WAIT_FOR_EVENT

22	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
21	Display Plane 1 C Vertical Blank Wait Enable <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait until the next Display Plane 1 C "Vertical Blank" event occurs. This event is described as the start of the next Display C vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event in the Device Programming Interface chapter of MI Functions.</p>	Format:	Enable
Format:	Enable		
20	Display Plane 6 Flip Pending Wait Enable <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 2 C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
19	Display Plane 12 Flip Pending Wait Enable <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 4 C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
18	Display Plane 11 Flip Pending Wait Enable <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 4 B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
17	Display Plane 10 Flip Pending Wait Enable <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 4 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
16	Display Plane 9 Flip Pending Wait Enable <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 3 C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
15	Display Plane 3 Flip Pending Wait Enable <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 1 C Flip Pending condition. If a flip</p>	Format:	Enable
Format:	Enable		

MI_WAIT_FOR_EVENT

	request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).		
14	<p>Display Plane 1 C Scan Line Wait Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait while a Display Plane 1 C "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register.</p>	Format:	Enable
Format:	Enable		
13:12	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ		
11	<p>Display Plane 1 B Vertical Blank Wait Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait until the next Display Plane 1 B "Vertical Blank" event occurs. This event is described as the start of the next Display B vertical blank period. Note that this can cause a wait for up to an entire refresh period.</p>	Format:	Enable
Format:	Enable		
10	<p>Display Plane 5 Flip Pending Wait Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 2 B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
9	<p>Display Plane 2 Flip Pending Wait Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
8	<p>Display Plane 1 B Scan Line Wait Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait while a Display Plane 1 B "Scan Line" condition exists. This condition is defined as the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register.</p>	Format:	Enable
Format:	Enable		
7	<p>Display Plane 8 Flip Pending Wait Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>This field enables a wait for the duration of a Display Plane 3 B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>	Format:	Enable
Format:	Enable		
6	<p>Display Plane 7 Flip Pending Wait Enable</p>		

MI_WAIT_FOR_EVENT

		Format:	Enable
		This field enables a wait for the duration of a Display Plane 3 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).	
5:4	Reserved		
		Format:	MBZ
3	Display Plane 1 A Vertical Blank Wait Enable		
		Format:	Enable
		This field enables a wait until the next Display Plane 1 A "Vertical Blank" event occurs. This event is described as the start of the next Display A vertical blank period. Note that this can cause a wait for up to an entire refresh period.	
2	Display Plane 4 Flip Pending Wait Enable		
		Format:	Enable
		This field enables a wait for the duration of a Display Plane 2 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).	
1	Display Plane 1 Flip Pending Wait Enable		
		Format:	Enable
		This field enables a wait for the duration of a Display Plane 1 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).	
0	Display Plane 1 A Scan Line Wait Enable		
		Format:	Enable
		This field enables a wait while a Display Plane 1 A "Scan Line" condition exists. This condition is defined as the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register.	

MI_WAIT_FOR_EVENT_2

MI_WAIT_FOR_EVENT_2		
Source:	RenderCS, BlitterCS	
Length Bias:	1	
<p>The MI_WAIT_FOR_EVENT_2 command is used to pause command stream processing of an engine (render or blitter) until a specific display event occurs (V-Blank) or a specific condition (Flip Pending, Scanline Pending) exists.</p> <ul style="list-style-type: none"> • Display engine can be configured to generate periodic V-Blank event to an engine. • An engine on executing MI_LOAD_SCANLINE_INCL/EXCL command for a display pipe, expects a corresponding scanline event response from display engine. Engine tracks the pending scanline response for each of the display pipe separately. • An engine on executing MI_DISPLAY_FLIP command for a display plane, expects a corresponding flip done event response from display engine. Engine tracks the pending flip done response (flip pending) for each of the display plane separately. Display flip could be of type Sync Flip or Async Flip, hence engine also tracks the type of flip (Sync or Async) along with the pending flip done response for a given display plane. <p>Only one event or condition can be specified in the command -- specifying multiple events or conditions is UNDEFINED. The command parser will halt until the event occurs or condition exists on parsing this command. Note that if a specified condition (Pending Flip Done response or Pending Scanline response) does not exist at the time the parser executes this command, the parser proceeds, treating this command as a no-operation (Ex: Command is no-operation when parsed with Display Plane Flip Pending Wait Enable set to Display Plane-1 when there is no outstanding flip done response for Display Plane-1 in the engine).</p> <p>Execution List Mode of Scheduling:</p> <p>An engine on evaluating unsuccessful MI_WAIT_FOR_EVENT_2 (results in pausing command stream) triggers synchronous context switch stating the switch reason in Context Status Buffer. With exception of not triggering synchronous context switch on unsuccessful MI_WAIT_FOR_EVENT_2 due to Flip Pending on an Async flip. Note that synchronous context switch can be inhibited through programming Inhibit Synchronous Context Switch bit in CTXT_SR_CTL register or by disabling arbitration through MI_ARB_ON_OFF command around MI_WAIT_FOR_EVENT_2. When synchronous context switch is inhibited and the engine is waiting on an unsuccessful MI_WAIT_FOR_EVENT_2, a submission of new execlist will trigger preemption process switching out the context. With exception of not triggering preemption on an unsuccessful MI_WAIT_FOR_EVENT_2 due to Flip Pending on an Async flip.</p> <p>Engine will always re-evaluate the wait condition for context switched out due to unsuccessful MI_WAIT_FOR_EVENT2 on resubmission of the context.</p>		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 0h MI_COMMAND
	Format: OpCode	
	28:23	MI Command Opcode
Default Value: 04h MI_WAIT_FOR_EVENT_2		
Format: OpCode		

MI_WAIT_FOR_EVENT_2

22:15	Reserved		MBZ
14:12	Display Pipe Scan Line Wait Enable	Format:	Enable
<p>This field enables a wait while a Display Pipe "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register.</p>			
		Value	Name
		0h	No Wait
		1h	Display Pipe A
		2h	Display Pipe B
		3h	Display Pipe C
		4h	Display Pipe D
		[5h,7h]	Reserved
11	Reserved	Format:	MBZ
10:8	Display Pipe Vertical Blank Wait Enable	Format:	Enable
<p>This field enables a wait until the next Display Pipe "Vertical Blank" event occurs. This event is described as the start of the next Display A vertical blank period. Note that this can cause a wait for up to an entire refresh period.</p>			
		Value	Name
		0h	No Wait
		1h	Display Pipe A
		2h	Display Pipe B
		3h	Display Pipe C
		4h	Display Pipe D
		[5h,7h]	Reserved
7:6	Reserved	Format:	MBZ
5:0	Display Plane Flip Pending Wait Enable	Format:	Enable
<p>This field enables a wait for the duration of a Display Plane Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).</p>			
		Value	Name
		0h	No Wait
		1h	Display Plane-1

MI_WAIT_FOR_EVENT_2

2h	Display Plane-2
3h	Display Plane-3
4h	Display Plane-4
5h	Display Plane-5
6h	Display Plane-6
7h	Display Plane-7
8h	Display Plane-8
9h	Display Plane-9
Ah	Display Plane-10
Bh	Display Plane-11
Ch	Display Plane-12
Dh	Display Plane-13
Eh	Display Plane-14
Fh	Display Plane-15
10h	Display Plane-16
11h	Display Plane-17
12h	Display Plane-18
13h	Display Plane-19
14h	Display Plane-20
15h	Display Plane-21
16h	Display Plane-22
17h	Display Plane-23
18h	Display Plane-24
19h	Display Plane-25
1Ah	Display Plane-26
1Bh	Display Plane-27
1Ch	Display Plane-28
1Dh	Display Plane-29
1Eh	Display Plane-30
1Fh	Display Plane-31
20h	Display Plane-32
[21h, 3Fh]	Reserved

Monitor Event

MSD_MONITOR_EVENT - Monitor Event								
Source:	EuSubFunctionGateway							
Length Bias:	1							
Gateway will record for this thread if this Event ID is signaled.								
DWord	Bit	Description						
0	31:29	Reserved Format: MBZ						
	28:25	Message Length Format: U4 Specifies the number of GRF registers sent as the message payload.						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>One [Default]</td> <td>See MDP_EVENT Event Data Payload definition.</td> </tr> </tbody> </table>	Value	Name	Description	1	One [Default]	See MDP_EVENT Event Data Payload definition.
		Value	Name	Description				
	1	One [Default]	See MDP_EVENT Event Data Payload definition.					
	24:20	Response Length Default Value: 0 None Format: U5 Specifies the number of GRF registers expected as the message response payload.						
	19:3	Reserved Format: MBZ						
	2:0	Monitor Event Subfunction Default Value: 0x2						
		Format: OpCode						

Monitor No Event

MSD_MONITOR_NO_EVENT - Monitor No Event								
Source:	EuSubFunctionGateway							
Length Bias:	1							
Gateway will stop recording any Events for this thread.								
DWord	Bit	Description						
0	31:29	Reserved Format: MBZ						
	28:25	Message Length Format: U4 Specifies the number of GRF registers sent as the message payload. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>One [Default]</td> <td>Data payload is ignored.</td> </tr> </tbody> </table>	Value	Name	Description	1	One [Default]	Data payload is ignored.
	Value	Name	Description					
	1	One [Default]	Data payload is ignored.					
	24:20	Response Length Default Value: 0 None Format: U5 Specifies the number of GRF registers expected as the message response payload.						
	19:3	Reserved Format: MBZ						
	2:0	Monitor No Event Subfunction Default Value: 0x3 Format: OpCode						

Move

mov - Move			
Source:	Eulsa		
Length Bias:	4		
<p>The mov instruction moves the components in src0 into the channels of dst. If src0 and dst are of different types, format conversion is performed. If src0 is a scalar immediate, the immediate value is loaded into enabled channels of dst. A mov with the same source and destination type, no source modifier, and no saturation is a raw move. A packed byte destination region (B or UB type with HorzStride == 1 and ExecSize > 1) can only be written using raw move.</p>			
<p>When denorm mode is flush to zero, a raw mov instruction with saturation modifier will not flush the denorm input or output to zero (Denorm is preserved).</p>			
<p>Format:</p> <pre>[(pred)] mov[.cmod] (exec_size) dst src0</pre>			
Programming Notes			
<p>A <i>mov</i> instruction with a source modifier always copies a denorm source value to a denorm destination value (in the manner of a raw move).</p>			
<p>There is no direct conversion from B/UB to DF or DF to B/UB. Use two instructions and a word or DWord intermediate type.</p>			
<p>There is no direct conversion from B/UB to Q/UQ or Q/UQ to B/UB. Use two instructions and a word or DWord intermediate integer type.</p>			
<p>There is no direct conversion from HF to DF or DF to HF. Use two instructions and F (Float) as an intermediate type.</p>			
<p>There is no direct conversion from HF to Q/UQ or Q/UQ to HF. Use two instructions and F (Float) or a word integer type or a DWord integer type as an intermediate type.</p>			
Restriction			
<p>Raw move is not supported for Float values in ALT mode if any values are infinities or NaNs.</p>			
<p>An accumulator can be a source or destination operand but not both.</p>			
<p>IP register must not be used as destination operand when EU Fusion is enabled.</p>			
Syntax			
<pre>[(pred)] mov[.cmod] (exec_size) reg reg [(pred)] mov[.cmod] (exec_size) reg imm32 [(pred)] mov[.cmod] (exec_size) reg imm64</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n]; }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

mov - Move

Src Types	Dst Types
*B,*W,*D	*B,*W,*D
*B,*W,*D	F
F	*B,*W,*D
F	F
*B,*W,*D	HF
F	HF
HF	*B,*W,*D
HF	F
HF	HF

DWord	Bit	Description
0..3	127:96	Src0.ImmValue[31:0] Exists If: $[[Src0.IsImm] == true]$
	95:92	CondCtrl Exists If: $[[Src0.IsImm] == false] \text{ OR } ([[Src0.DataType] != :q] \text{ AND } [[Src0.DataType] != :uq] \text{ AND } [[Src0.DataType] != :df])$ Format: FlagModifier
	95:64	Src0.ImmValue[63:32] Exists If: $[[Src0.IsImm] == true] \text{ AND } ([[Src0.DataType] == :q] \text{ OR } [[Src0.DataType] == :uq] \text{ OR } [[Src0.DataType] == :df])$
	87:84	Src0.VertStride Exists If: $[[Src0.IsImm] == false] \text{ OR } ([[Src0.DataType] != :q] \text{ AND } [[Src0.DataType] != :uq] \text{ AND } [[Src0.DataType] != :df])$ Format: VertStride
	83:81	Src0.Width Exists If: $[[Src0.IsImm] == false] \text{ OR } ([[Src0.DataType] != :q] \text{ AND } [[Src0.DataType] != :uq] \text{ AND } [[Src0.DataType] != :df])$ Format: Width
	80	Src0.AddrMode Exists If: $[[Src0.IsImm] == false] \text{ OR } ([[Src0.DataType] != :q] \text{ AND } [[Src0.DataType] != :uq] \text{ AND } [[Src0.DataType] != :df])$ Format: AddrMode
	79:66	Src0.Operand Exists If: $((([Src0.IsImm] == false) \text{ OR } ([[Src0.DataType] != :q] \text{ AND } [[Src0.DataType] != :uq] \text{ AND } [[Src0.DataType] != :df])) \text{ AND } [[Src0.AddrMode] == Direct])$ Format: DirectOperand
	79:66	Src0.Operand

mov - Move

		Exists If:	(([[Src0.IsImm]]==false) OR (([[Src0.DataType]]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)
		Format:	IndirectOperand
65:64	Src0.HorzStride		
		Exists If:	(([[Src0.IsImm]]==false) OR (([[Src0.DataType]]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
		Format:	HorzStride
63:50	Dst.Operand		
		Exists If:	(([[Dst.AddrMode]]==Indirect)
		Format:	IndirectOperand
63:50	Dst.Operand		
		Exists If:	(([[Dst.AddrMode]]==Direct)
		Format:	DirectOperand
49:48	Dst.HorzStride		
		Format:	HorzStride
47	Reserved		
		Format:	MBZ
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
		Format:	SrcMod
43:40	Src0.DataType		
		Exists If:	(([[Src0.IsImm]]==false)
		Format:	RegDataType
43:40	Src0.DataType		
		Exists If:	(([[Src0.IsImm]]==true)
		Format:	ImmDataType
39:36	Dst.DataType		
		Format:	RegDataType
35	Dst.AddrMode		
		Format:	AddrMode
34	Saturate		
		Format:	Saturate

mov - Move

33	AccWrCtrl Format: AccWrCtrl										
32	AtomicCtrl Format: AtomicCtrl										
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>		Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.									
30	Reserved										
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>		Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the										

mov - Move			
	content of the selected flag register.		
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Move Indexed

movi - Move Indexed	
Source:	Eulsa
Length Bias:	4
<p>The movi instruction performs a fast component-wise indexed move for subfields from src0 to dst. The sourceoperand must be an indirectly-addressed register. All channels of the source operand share the same registernumber, which is provided by the register field of the first address subregister, with a possible immediateregister offset. The register fields of the subsequent address subregisters are ignored by hardware. Thesubregister number of a source channel is provided by the subregister field of the corresponding addresssubregister, with a possible immediate subregister offset.</p> <p>The destination register may be either a directly-addressed or an indirectly-addressed register. This instruction effectively performs a subfield shuffling from one register to another.</p>	
<p>Format:</p> <pre>[(pred)] movi (exec_size) dst src0 src1</pre>	
Programming Notes	
<p>HW Implementation Details:</p> <p>The source register is calculated by adding the register portion of the first index register with the register portion of the address immediate, $a0.0[11:5] + addr_imm[9:5]$</p> <p>For byte movi, byte0 of the destination is selected by $(a0.0[4:0])$, byte1 is selected by $(a0.1[4:0])$, ..., and byte7 is selected by $(a0.7[4:0])$. The rest of the bytes are undefined.</p> <p>For word movi, byte0 of the destination is selected by $(a0.0[4:1] \& 0)$, byte1 is selected by $(a0.0[4:1] \& 1)$, byte2 is selected by $(a0.1[4:1] \& 0)$, byte3 is selected by $(a0.1[4:1] \& 1)$, ..., and byte15 is selected by $(a0.7[4:1] \& 1)$. The rest of the bytes are undefined.</p> <p>For DWord or float movi, byte0 of the destination is selected by $(a0.0[4:2] \& 00b)$, byte1 is selected by $(a0.0[4:2] \& 01b)$, byte2 is selected by $(a0.0[4:2] \& 10b)$, byte3 is selected by $(a0.0[4:2] \& 11b)$, byte4 is selected by $(a0.1[4:2] \& 00b)$, byte5 is selected by $(a0.1[4:2] \& 01b)$, ..., byte31 is selected by $(a0.7[4:2] \& 11b)$.</p> <p>For all 3 conditions above, $a0.n[4:0] = a0.n[4:0] + addr_imm[4:0]$.</p>	
Restriction	
Source operand cannot be accumulators. The source operand must be a general register.	
The source and destination must have the same type.	
The address register for the source must be a0.0 or a0.8.	
The destination register (directly or indirectly addressed) must be 16-byte aligned.	
The destination region (directly or indirectly addressed) must point to the same GRF register.	
The destination stride in bytes must equal the source element size in bytes.	
All the index registers (address subregisters) used must point to the same GRF register.	
The instruction must use 1x1 indirect regioning.	
The destination offset is only used to create channel enables. Each element of the destination is directly mapped to the index registers for the movi instruction. i.e. a0.0 -> dst.0, a0.1 -> dst.1, a0.2 -> dst.2, etc.	
Only 8 address subregisters are used (a0.0-a0.7 or a0.8-a0.15). Destination element will be sourced from	

movi - Move Indexed

address register (a0.0 or a0.8), for example:

```

movi (8) r31.0:uw r[a0.0,0]<1;1,0>:uw // r31.0:uw<-a0.0:uw, r31.1:uw<-a0.1:uw, etc.
movi (8) r31.0:uw r[a0.8,0]<1;1,0>:uw // r31.0:uw<-a0.8:uw, r31.1:uw<-a0.9:uw, etc.
movi (8) r31.8:uw r[a0.0,0]<1;1,0>:uw // r31.8:uw<-a0.0:uw, r31.9:uw<-a0.1:uw, etc.
movi (8) r31.8:uw r[a0.8,0]<1;1,0>:uw // r31.8:uw<-a0.8:uw, r31.9:uw<-a0.9:uw, etc.
movi (8) r31.0:ud r[a0.0,0]<1;1,0>:ud // r31.0:ud<-a0.0:ud, r31.1:ud<-a0.1:ud, etc.
movi (8) r31.0:ud r[a0.8,0]<1;1,0>:ud // r31.0:ud<-a0.8:ud, r31.1:ud<-a0.9:ud, etc.
    
```

Conditional Modifier is not allowed for this instruction.

Syntax

```

[(pred)] movi (exec_size) reg reg null
[(pred)] movi (exec_size) reg reg imm32
    
```

Pseudocode

```

Evaluate (WrEn);
    srcregfile = regfile(src0);
    imm_offset = (src1 == NULL) ? addr_imm : src1;
    srcregbase = reg(address[0]) + reg(imm_offset);
    for ( n = 0; n < RegWidth; n++ ) {
        if ( WrEn.chan[n] ) {
            srcsubreg = subreg(address[n] + imm_offset);
            dst.chan[n] = srcregfile.srcreg.srcsubreg;
        }
    }
    
```

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	Y	Y

Src Types	Dst Types
B	B
UB	UB
W	W
UW	UW
D	D
UD	UD
F	F

DWord	Bit	Description
0..3	127:96	Src0.ImmValue[31:0] Exists If: $[[\text{Src0.IsImm}] = \text{true}]$
	95:92	CondCtrl Exists If: $[[\text{Src0.IsImm}] = \text{false}] \text{ OR } ((([\text{Src0.DataType}] != \text{:q}) \text{ AND } ([\text{Src0.DataType}] != \text{:uq}) \text{ AND } ([\text{Src0.DataType}] != \text{:df})$
		Format: FlagModifier

movi - Move Indexed

95:64	Src0.ImmValue[63:32]	
	Exists If:	(([Src0.IsImm]==true) AND ((([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df)))
	Src0.VertStride	
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
	Format:	VertStride
	Src0.Width	
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
	Format:	Width
	Src0.AddrMode	
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
Format:	AddrMode	
79:66	Src0.Operand	
	Exists If:	((([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct))
Format:	DirectOperand	
79:66	Src0.Operand	
	Exists If:	((([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect))
Format:	IndirectOperand	
65:64	Src0.HorzStride	
	Exists If:	(([Src0.IsImm]==false) OR ((([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)))
Format:	HorzStride	
63:50	Dst.Operand	
	Exists If:	(([Dst.AddrMode]==Indirect))
Format:	IndirectOperand	
63:50	Dst.Operand	
	Exists If:	(([Dst.AddrMode]==Direct))
Format:	DirectOperand	
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Reserved	
	Format:	MBZ

movi - Move Indexed

46	Src0.IsImm This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
45:44	Src0.Mod Format: SrcMod	
43:40	Src0.DataType Exists If: ([Src0.IsImm]==false) Format: RegDataType	
43:40	Src0.DataType Exists If: ([Src0.IsImm]==true) Format: ImmDataType	
39:36	Dst.DataType Format: RegDataType	
35	Dst.AddrMode Format: AddrMode	
34	Saturate Format: Saturate	
33	AccWrCtrl Format: AccWrCtrl	
32	AtomicCtrl Format: AtomicCtrl	
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
	0	Normal [Default]
	1	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations	

movi - Move Indexed

supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.			
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff Format: ChanOff This field provides offset information for ARF selection. This can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.		
18:16	ExecSize Format: ExecSize		

movi - Move Indexed						
		This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.				
	15:0	<table border="1" style="width: 100%;"> <tr> <td colspan="2">Header</td> </tr> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: center;">Header</td> </tr> </table>	Header		Format:	Header
Header						
Format:	Header					

Multiply

mul - Multiply			
Source:	Eulsa		
Length Bias:	4		
<p>The mul instruction performs component-wise multiplication of src0 and src1 and stores the results in dst. When multiplying integer datatypes, if src0 is DW and src1 is W, irrespective of the destination datatype, the accumulator maintains full 48-bit precision. This is required to handle the macro for 32x32 multiplication. The macro described in the mach instruction should be used to obtain the full precision 64-bit multiplication results.</p> <p>Note: A 32x32 multiply operation is handled natively, without a macro. When operating in this mode, the resulting 64-bit data is packed, unlike the macro, where the lower and upper 32 bits of the result are written to different general registers by two separate instructions. Refer to the macro description for details.</p> <p>When multiplying integer data types, if one of the sources is a DW, the resulting full precision data is stored in the accumulator. However, if the destination data type is either W or DW, the low bits of the result are written to the destination register and the remaining high bits are discarded. This results in undefined Overflow and Sign flags. Therefore, conditional modifiers and saturation (.sat) cannot be used in this case.</p>			
Format:	<pre>[(pred)] mul[.cmod] (exec_size) dst src0 src1</pre>		
Restriction			
Integer source operands cannot be accumulators.			
When multiplying a DW and any lower precision integer, the DW operand must on src0.			
When multiplying a DW and any lower precision integer, source modifier is not supported.			
Syntax			
<pre>[(pred)] mul[.cmod] (exec_size) reg reg reg [(pred)] mul[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn);for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] * src1.chan[n]; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
*B	*B		
*B	*W		
*B	*D		
*W	*W		
*W	*D		

mul - Multiply

F	F		
HF	HF		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
	125:122	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	121:120	Src1.Mod	
		Exists If:	([Src1.IsImm]==false)
		Format:	SrcMod
	119:116	Src1.VertStride	
		Exists If:	([Src1.IsImm]==false)
		Format:	VertStride
	115:113	Src1.Width	
Exists If:		([Src1.IsImm]==false)	
Format:		Width	
112	Src1.AddrMode		
	Exists If:	([Src1.IsImm]==false)	
	Format:	AddrMode	
111:98	Src1.Operand		
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)	
	Format:	DirectOperand	
111:98	Src1.Operand		
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)	
	Format:	IndirectOperand	
97:96	Src1.HorzStride		
	Exists If:	([Src1.IsImm]==false)	
	Format:	HorzStride	
95:92	CondCtrl		
	Format:	FlagModifier	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==false)	

mul - Multiply

	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true

mul - Multiply

45:44	Src0.Mod	Format:	SrcMod
43:40	Src0.DataType	Exists If:	((Src0.IsImm) == false)
		Format:	RegDataType
43:40	Src0.DataType	Exists If:	((Src0.IsImm) == true)
		Format:	ImmDataType
39:36	Dst.DataType	Format:	RegDataType
35	Dst.AddrMode	Format:	AddrMode
34	Saturate	Format:	Saturate
33	AccWrCtrl	Format:	AccWrCtrl
32	AtomicCtrl	Format:	AtomicCtrl
31	MaskCtrl	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
		Value	Name
		Description	
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
		Value	Name
		Description	
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.

mul - Multiply

	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	<p>PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>			Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										
27:24	<p>PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>											
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>											
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>											
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>			Format:	ChanOff							
Format:	ChanOff											
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>			Format:	ExecSize							
Format:	ExecSize											
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>			Format:	Header							
Format:	Header											

Multiply Accumulate

mac - Multiply Accumulate			
Source:	Eulsa		
Length Bias:	4		
The mac instruction takes component-wise multiplication of src0 and src1, adds the results with the corresponding accumulator values, and then stores the final results in dst.			
Format:	[(pred)] mac[.cmod] (exec_size) dst src0 src1		
Programming Notes			
When source and destination datatypes are different, the implied datatype for the accumulator operand is always the destination datatype.			
Restriction			
The conditional modifier and saturation (.sat) must not be used when src0 or src1 are dwords.			
Syntax			
[(pred)] mac[.cmod] (exec_size) reg reg reg [(pred)] mac[.cmod] (exec_size) reg reg imm32			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] * src1.chan[n] + acc0.chan[n]; }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
*B,*W	*B,*W,*D		
F	F		
HF	HF		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
	125:122	Reserved	
Exists If:		([Src1.IsImm]==false)	
Format:	MBZ		

mac - Multiply Accumulate

121:120	Src1.Mod	
	Exists If:	([Src1.IsImm]==false)
	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	([Src1.IsImm]==false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	([Src1.IsImm]==false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	

mac - Multiply Accumulate

	Exists If:	([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Src1.IsImm		
	This field indicate that Source 1 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==false)	
	Format:	RegDataType	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==true)	
	Format:	ImmDataType	
39:36	Dst.DataType		
	Format:	RegDataType	
35	Dst.AddrMode		

mac - Multiply Accumulate

	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
		Description
	0	Normal [Default]
		Normal. Per channel write enable used for final write enable generation.
	1	NoMask
		NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name
		Description
	0	NoCompaction [Default]
		No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted
		Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name
		Description
	0	Positive [Default]
		Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative
		Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

mac - Multiply Accumulate

27:24	<p>PredCtrl</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl
Format:	PredCtrl		
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Multiply Accumulate High

mach - Multiply Accumulate High	
Source:	Eulsa
Length Bias:	4
<p>The mach instruction performs DWord integer multiply-accumulate operation and outputs the high DWord (bits 63:32). For each enabled channel, this instruction multiplies the DWord in src0 with the high word of the DWord in src1, left shifts the result by 16 bits, adds it with the corresponding accumulator values, and keeps the whole 64-bit result in the accumulator. It then stores the high DWord (bits 63:32) of the results in dst. This instruction is intended to be used to emulate 32-bit DWord integer multiplication by using the large number of bits available in the accumulator. For example, the following instructions perform vector multiplication of two 32-bit signed integer sources from r2 and r3 and store the resulting vectors with the high 32 bits in r5 and the low 32 bits in r6.</p> <pre>mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d mov (8) r6.0<1>:d acc0:d // Low 32 bits.</pre> <p>Here is a different example including negation. An added preliminary mov is required for source modification on src1.</p> <pre>mov (8) r3.0<1>:d -r3<8;8,1>:d mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d // High 32 bits mov (8) r6.0<1>:d acc0:d // Low 32 bits.</pre> <p>The mach should have channel enable from the destHI of IMUL, the mov should have the channel enable from the destLO of IMUL. As mach is used to generate part of the 64-bit DWord integer results, saturation modifier should not be used. In fact, saturation modifier should not be used for any of these four instructions. Source and destination operands must be DWord integers. Source and destination must be of the same type, signed integer or unsigned integer. If dst is UD, src0 and src1 may be UD and/or D. However, if any of src0 and src1 is D, source modifier (abs) must be present to convert it to match with dst. If dst is D, src0 and src1 must also be D. They cannot be UD as it may cause unexpected overflow because the computed results are limited to 64 bits.</p>	
<p>Format:</p> <pre>[(pred)] mach[.cmod] (exec_size) dst src0 src1</pre>	
Restriction	
Accumulator is an implicit source and thus cannot be an explicit source operand.	
The accumulator is an implicit destination and thus cannot be an explicit destination operand.	
Syntax	
<pre>[(pred)] mach[.cmod] (exec_size) reg reg reg [(pred)] mach[.cmod] (exec_size) reg reg imm32</pre>	
Pseudocode	
<pre>Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) {</pre>	

mach - Multiply Accumulate High

```

temp.chan[n][63:0] = (src1.chan[n][31:16] *
    src0.chan[n][31:0]) < 16 + acc.chan[n][63:0];
if (AccWrEn) {
    acc.chan[n][63:0] = temp.chan[n][63:0];
    dst.chan[n][31:0] = temp.chan[n][63:32];
}
else {
    dst.chan[n][31:0] = temp.chan[n][31:0];
}
}
}

```

Errata	Description
	A source modifier must not be used on src1 for the macro operation. This applies to both mul and mach of the macro. If source modifier is required, an additional mov instruction may be used before the macro.

Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	Y

Src Types	Dst Types
D	D
UD	UD

DWord	Bit	Description
0..3	127:126	Reserved
		Exists If: ([Src1.IsImm]==false)
		Format: MBZ
	127:96	Src1.ImmValue[31:0]
		Exists If: ([Src1.IsImm]==true)
	125:122	Reserved
		Exists If: ([Src1.IsImm]==false)
		Format: MBZ
	121:120	Src1.Mod
		Exists If: ([Src1.IsImm]==false) Format: SrcMod
	119:116	Src1.VertStride
		Exists If: ([Src1.IsImm]==false) Format: VertStride
	115:113	Src1.Width
		Exists If: ([Src1.IsImm]==false) Format: Width
112	Src1.AddrMode	
	Exists If: ([Src1.IsImm]==false)	

mach - Multiply Accumulate High		
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	(([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct))
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	(([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect))
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	(([Src1.IsImm]==false))
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	(([Src1.IsImm]==false))
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	(([Src1.IsImm]==true))
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	(([Src0.AddrMode]==Direct))
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	(([Src0.AddrMode]==Indirect))
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	(([Dst.AddrMode]==Direct))
	Format:	DirectOperand
63:50	Dst.Operand	

mach - Multiply Accumulate High

	Exists If:	(([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	(([Src0.IsImm]==false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	(([Src0.IsImm]==true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	

mach - Multiply Accumulate High

Value	Name	Description
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
Format:		MBZ
<p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p>		
Value	Name	Description
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
<p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p>		
Value	Name	Description
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl	
Format:		PredCtrl
<p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		
23	FlagRegNum[0]	
<p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	FlagSubRegNum	
<p>This field specifies the sub-register number for a flag register operand. There are two sub-</p>		

mach - Multiply Accumulate High			
	<p>registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Multiply Add

mad - Multiply Add			
Source:	Eulsa		
Length Bias:	4		
Description			
<p>The mad instruction takes component-wise multiplication of src1 and src2, adds the results with the corresponding src0 values, and then stores the final results in dst.</p> <p>The conditional modifier and saturation (.sat) must not be used when src1 or src2 are dwords.</p> <p>Plane and Linear Interpolation instructions are removed. The following macros must be used to emulate Plane and Linear Interpolation operations.</p> <p>Plane Instruction Emulation The below plane instruction <code>pln (16) r20.0<1>:f r10.4<0;1,0>:f r4.0<8;8,1>:fis emulated as below</code> <code>mad (8) acc0<1>:f r10.7<0;1,0>:f r4.0<8;8,1>:f r10.4<0;1,0>:fmad (8) r20.0<1>:f acc0<8;8,1>:f r5.0<8;8,1>:f r10.5<0;1,0>:fmad (8) acc0<1>:f r10.7<0;1,0>:f r6.0<8;8,1>:f r10.4<0;1,0>:fmad (8) r21.0<1>:f acc0<8;8,1>:f r7.0<8;8,1>:f r10.5<0;1,0>:f</code> In case of SIMD8 pln instruction only the first pair of mad instructions are used.</p> <p>Linear Interpolation Instruction Emulation The below lrp instruction <code>lrp (16) r40.0<1>:f r10.0<8;8,1>:f r20.0<8;8,1>:f r30.0<8;8,1>:fis emulated as below</code> <code>mad (8) acc0<1>:f r30.0<8;8,1>:f r10.0<8;8,1>:f r20.0<8;8,1>:fmad (8) r40.0<1>:f acc0<8;8,1>:f -r10.0<8;8,1>:f r30.0<8;8,1>:fmad (8) acc0<1>:f r31.0<8;8,1>:f r11.0<8;8,1>:f r21.0<8;8,1>:fmad (8) r41.0<1>:f acc0<8;8,1>:f -r11.0<8;8,1>:f r31.0<8;8,1>:f</code> In case of SIMD8 lrp instruction only the first pair of mad instructions are used.</p>			
Format: <code>[(pred)] mad[.cmod] (exec_size) dst src0 src1 src2</code>			
Restriction			
Src1/Src2 for Integer source operands cannot be accumulators. Src0 is allowed to use accumulator.			
When multiplying a DW and any lower precision integer, source modifier is not supported.			
All three-source instructions have certain restrictions, described in Instruction Formats.			
Syntax			
<code>[(pred)] mad[.cmod] (exec_size) reg reg reg reg [(pred)] mad[.cmod] (exec_size) reg reg reg imm16 [(pred)] mad[.cmod] (exec_size) reg imm16 reg reg</code>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src1.chan[n] * src2.chan[n] + src0.chan[n]; }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types		Dst Types	
F		F	
HF		HF	
B		W	

mad - Multiply Add

W, D		W, D		
DWord	Bit	Description		
0..3	127:114	Src2.Operand		
		Exists If:	(([Src2.IsImm]==false) AND ([Header][Opcode]!=madm))	
			Format:	DirectOperand
	127:114	Src2.Operand		
		Exists If:	(([Src2.IsImm]==false) AND ([Header][Opcode]==madm))	
			Format:	MacroOperand
	127:112	Src2.ImmValue[15:0]		
		Exists If:	([Src2.IsImm]==true)	
	113:112	Src2.HorzStride		
		Exists If:	([Src2.IsImm]==false)	
			Format:	HorzStride
	111:98	Src1.Operand		
		Exists If:	([Header][Opcode]!=madm)	
			Format:	DirectOperand
	111:98	Src1.Operand		
		Exists If:	([Header][Opcode]==madm)	
			Format:	MacroOperand
	97:96	Src1.HorzStride		
Format:		HorzStride		
95:92	CondCtrl			
	Format:	FlagModifier		
91	Src1.VertStride[1]			
	Format:	TernaryVertStride[1]		
90:88	Src1.DataType			
	Format:	TernaryDataType		
87:86	Src1.Mod			
	Format:	SrcMod		
85:84	Src2.Mod			
	Format:	SrcMod		
83	Src1.VertStride[0]			
	Format:	TernaryVertStride[0]		
82:80	Src2.DataType			
	Format:	TernaryDataType		
79:66	Src0.Operand			

mad - Multiply Add

	Exists If:	([Src0.IsImm] == false) AND ([Header][Opcode] != madm)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm] == false) AND ([Header][Opcode] == madm)	
	Format:	MacroOperand	
79:64	Src0.ImmValue[15:0]		
	Exists If:	([Src0.IsImm] == true)	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm] == false)	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Header][Opcode] != madm)	
	Format:	DirectOperand	
Programming Notes			
The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,			
63:50	Dst.Operand		
	Exists If:	([Header][Opcode] == madm)	
	Format:	MacroOperand	
49	Reserved		
	Format:	MBZ	
48	Dst.HorzStride		
	This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand.		
	Value	Name	
	0	1 element	
	1	2 element	
47	Src2.IsImm		
	This field indicate that Source 2 operand is carrying an immediate value.		
	Value	Name	
	0	false	
	1	true	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
	1	true	

mad - Multiply Add

45:44	Src0.Mod	
	Format:	SrcMod
43	Src0.VertStride[1]	
	Format:	TernaryVertStride[1]
42:40	Src0.DataType	
	Format:	TernaryDataType
39	ExecDataType	
	This field indicate the datatype mode of ternary instruction. Integer or Float.	
	Value	Name
	0	Integer
	1	Float
38:36	Dst.DataType	
	Format:	TernaryDataType
35	Src0.VertStride[0]	
	Format:	TernaryVertStride[0]
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
	0	Normal [Default]
	1	NoMask
	Normal. Per channel write enable used for final write enable generation.	
	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	

mad - Multiply Add

Value	Name	Description		
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.		
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.		
28	PredInv	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
Value	Name	Description		
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.		
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.		
27:24	PredCtrl	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl
Format:	PredCtrl			
23	FlagRegNum[0]	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff			
18:16	ExecSize	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize			
15:0	Header	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header			



No Operation

nop - No Operation			
Source:	Eulsa		
Length Bias:	4		
Do nothing. The nop instruction takes an instruction dispatch but performs no operation. It can be used for assembly patching in memory, or to insert a delay in the program sequence.			
Format:	nop		
Restriction			
The nop instruction takes no instruction options other than Breakpoint.			
Syntax			
nop			
Pseudocode			
{ ; // The null statement, which does nothing. }			
Predication	Conditional Modifier	Saturation	Source Modifier
N	N	N	N
DWord	Bit	Description	
0..3	127:31	Reserved Format: <input type="text"/> MBZ	
	30	Reserved	
	29:28	Reserved Format: <input type="text"/> MBZ	
	27:26	Reserved Format: <input type="text"/> MBZ	
	25:18	Reserved Format: <input type="text"/> MBZ	
	17:16	Reserved Format: <input type="text"/> MBZ	
	15:0	Header Format: <input type="text"/> Header	

Oword Aligned Block Read MSD

MSD0R_OWAB - Oword Aligned Block Read MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18	Legacy Message Default Value: 0h Format: Opcode Legacy Message
	17:14	Message Type Default Value: 01h Format: Opcode Aligned Block Read message
	13	Reserved Format: MBZ
	12:11	Reserved Format: MBZ
	10:8	Data Elements Format: MDC_DB_OW Specifies the number of contiguous Owords to be read

MSD0R_OWAB - Oword Aligned Block Read MSD			
7:0	<p>Binding Table Index</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_BTS_A32</td> </tr> </table> <p>Specifies the Binding Table Index for the message</p>	Format:	MDC_BTS_A32
Format:	MDC_BTS_A32		

Oword Block Read MSD

MSDOR_OWB - Oword Block Read MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18	Legacy Message Default Value: 0h Format: Opcode Legacy Message
	17:14	Message Type Default Value: 00h Format: Opcode Block Read message
	13	Block Message Subtype Default Value: 0 Format: Opcode Oword Block Read/Write subtype
	12:11	Reserved Format: MBZ
	10:8	Data Elements

MSD0R_OWB - Oword Block Read MSD							
	<table border="1"> <tr> <td>Format:</td> <td>MDC_DB_OW</td> </tr> <tr> <td colspan="2">Specifies the number of contiguous Owords to be read or written</td> </tr> </table>	Format:	MDC_DB_OW	Specifies the number of contiguous Owords to be read or written			
Format:	MDC_DB_OW						
Specifies the number of contiguous Owords to be read or written							
7:0	<table border="1"> <tr> <td colspan="2">Binding Table Index</td> </tr> <tr> <td>Format:</td> <td>MDC_BTS_SLM_A32</td> </tr> <tr> <td colspan="2">Specifies the Binding Table Index for the message</td> </tr> </table>	Binding Table Index		Format:	MDC_BTS_SLM_A32	Specifies the Binding Table Index for the message	
Binding Table Index							
Format:	MDC_BTS_SLM_A32						
Specifies the Binding Table Index for the message							

Oword Block Write MSD

MSD0W_OWB - Oword Block Write MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header.
	18	Legacy Message Default Value: 0h Format: Opcode Legacy Message
	17:14	Message Type Default Value: 08h Format: Opcode Block Write message
	13	Block Message Subtype Default Value: 0 Format: Opcode Oword Block subtype
	12:11	Reserved Format: MBZ
	10:8	Data Elements

MSD0W_OWB - Oword Block Write MSD							
	<table border="1"> <tr> <td>Format:</td> <td>MDC_DB_OW</td> </tr> <tr> <td colspan="2">Specifies the number of contiguous Owords to be read or written</td> </tr> </table>	Format:	MDC_DB_OW	Specifies the number of contiguous Owords to be read or written			
Format:	MDC_DB_OW						
Specifies the number of contiguous Owords to be read or written							
7:0	<table border="1"> <tr> <td colspan="2">Binding Table Index</td> </tr> <tr> <td>Format:</td> <td>MDC_BTS_SLM_A32</td> </tr> <tr> <td colspan="2">Specifies the Binding Table Index for the message</td> </tr> </table>	Binding Table Index		Format:	MDC_BTS_SLM_A32	Specifies the Binding Table Index for the message	
Binding Table Index							
Format:	MDC_BTS_SLM_A32						
Specifies the Binding Table Index for the message							

PIPE_CONTROL

PIPE_CONTROL			
Source:	RenderCS		
Length Bias:	2		
The PIPE_CONTROL command is used to effect the synchronization described above.			
Programming Notes			
<p>SW must follow below programming restrictions when programming PIPECONTROL command for POCS:</p> <ul style="list-style-type: none"> • Write cache flush bits must not be set (Render Target Cache Flush Enable, DC Flush Enable, Depth Cache Flush Enable) • Post Sync Operations must not be set to Write PS Depth Count • Stall at Pixel Scoreboard must not be set • Notify Enable must not be set. • Depth Stall Enable must not be set. • Generic Media State Clear must not be set. • PSD Sync Enable must not be set. 			
<p>SW must follow below programming restrictions when programming PIPE_CONTROL command for ComputeCS:</p> <ul style="list-style-type: none"> • "Command Streamer Stall Enable" must be always set. • Post Sync Operations must not be set to Write PS Depth Count • Following bits must not be set when programmed for ComputeCS <ul style="list-style-type: none"> • "Render Target Cache Flush Enable", "Depth Cache Flush Enable" and "Tile Cache Flush Enable" • "Depth Stall Enable", Stall at Pixel Scoreboard and "PSD Sync Enable". • "OVR Tile 0 Flush", "TBIMR Force Batch Closure", "AMFS Flush Enable" "VF Cache Invalidation Enable" and "Global Snapshot Count Reset". 			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	2h PIPE_CONTROL
		Format:	OpCode
	23:16	3D Command Sub Opcode	

PIPE_CONTROL						
		<table border="1"> <tr> <td>Default Value:</td> <td>0h PIPE_CONTROL</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h PIPE_CONTROL	Format:	OpCode
	Default Value:	0h PIPE_CONTROL				
	Format:	OpCode				
	15:12	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ				
	11	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ				
	10	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ				
	9	HDC Pipeline Flush If set, HDC and LSC ensures its pipeline is flushed and the memory transactions are coherent in L3\$ as part of the flush operation. The HDC read-only cache is also flushed as well. This will not result in flushing L3\$ portion that caches DC writes.				
8	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
7:0	DWord Length <table border="1"> <tr> <td>Default Value:</td> <td>4h DWORD_COUNT_n</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table> Total Length - 2. Excludes DWord (0,1).	Default Value:	4h DWORD_COUNT_n	Format:	=n	
Default Value:	4h DWORD_COUNT_n					
Format:	=n					
1	31	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
	Format:	MBZ				
30	L3 Fabric Flush <table border="1"> <thead> <tr> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td> <p>Pipelined L3 Fabric Flush:</p> <p>When Depth Stall Enable is set L3 Fabric Flush is pipelined along with the workload and issued from raster stage in the pipeline. Flush marker for this flush type on reaching raster stage, will ensure all the prior workload is complete and then L3 Fabric Flush is performed before allowing the workload following the flush marker to execute.</p> <p>By default L3 Fabric Flush happens top of the pipe as part of post-synchronization operation on a flush completion. This bit must be programmed for all contexts and hence it can be initialized during golden context creation time.</p> <p>Usages:</p> <p>This mechanism is used to flush the updated compressed control state of an surface during fast clear to global observable point before the rendering operations are started using these surfaces. Most common usage case is to flush the L1 (Color, Depth) caches with L3 Fabric Flush and Depth Stall Enable post Fast Clears prior to starting the rendering operations. Refer Render Target Fast Clear and Depth Buffer Clear sections for more details.</p> <ul style="list-style-type: none"> - For a sequence of color fast clears </td> </tr> </tbody> </table>	Description	<p>Pipelined L3 Fabric Flush:</p> <p>When Depth Stall Enable is set L3 Fabric Flush is pipelined along with the workload and issued from raster stage in the pipeline. Flush marker for this flush type on reaching raster stage, will ensure all the prior workload is complete and then L3 Fabric Flush is performed before allowing the workload following the flush marker to execute.</p> <p>By default L3 Fabric Flush happens top of the pipe as part of post-synchronization operation on a flush completion. This bit must be programmed for all contexts and hence it can be initialized during golden context creation time.</p> <p>Usages:</p> <p>This mechanism is used to flush the updated compressed control state of an surface during fast clear to global observable point before the rendering operations are started using these surfaces. Most common usage case is to flush the L1 (Color, Depth) caches with L3 Fabric Flush and Depth Stall Enable post Fast Clears prior to starting the rendering operations. Refer Render Target Fast Clear and Depth Buffer Clear sections for more details.</p> <ul style="list-style-type: none"> - For a sequence of color fast clears 			
Description						
<p>Pipelined L3 Fabric Flush:</p> <p>When Depth Stall Enable is set L3 Fabric Flush is pipelined along with the workload and issued from raster stage in the pipeline. Flush marker for this flush type on reaching raster stage, will ensure all the prior workload is complete and then L3 Fabric Flush is performed before allowing the workload following the flush marker to execute.</p> <p>By default L3 Fabric Flush happens top of the pipe as part of post-synchronization operation on a flush completion. This bit must be programmed for all contexts and hence it can be initialized during golden context creation time.</p> <p>Usages:</p> <p>This mechanism is used to flush the updated compressed control state of an surface during fast clear to global observable point before the rendering operations are started using these surfaces. Most common usage case is to flush the L1 (Color, Depth) caches with L3 Fabric Flush and Depth Stall Enable post Fast Clears prior to starting the rendering operations. Refer Render Target Fast Clear and Depth Buffer Clear sections for more details.</p> <ul style="list-style-type: none"> - For a sequence of color fast clears 						

PIPE_CONTROL

- A single PIPE_CONTROL command with Render Target Cache Flush, L3 Fabric Flush and Depth Stall set at the end of the sequence suffices. This assumes the output of the fast clears are used only post this PIPE_CONTROL command.
- For a sequence of depth fast clears
- A single PIPE_CONTROL command with Depth Cache Flush, L3 Fabric Flush and Depth Stall set at the end of the sequence suffices. This assumes the output of the depth fast clears are used only post this PIPE_CONTROL command.
- For a sequence of mixed color/depth fast clears.
- A single PIPE_CONTROL command with Depth Cache Flush, Render Target Cache Flush, L3 Fabric Flush and Depth Stall set at the end of the sequence suffices. This assumes the output of the color/depth fast clears are used only post this PIPE_CONTROL command.

Value	Name	Description
1		HW will do a L3 Fabric Flush on completion of flush for the corresponding PIPECONTROL. Setting this bit will force HW to send a marker downstream for a flush completion.
0		HW will not force a "L3 Fabric Flush" on completion of flush and will only do on need basis.

29 **Command Cache Invalidate Enable**

Format:	Enable
---------	--------

When set the command cache for commands parsed at the top of the pipe will be invalidated. This bit is independent from the other bits in this command and will be executed prior to the pipeline being flushed.

28 **Tile Cache Flush Enable**

Setting this bit will force Tile Cache (contains both color and depth data) to be flushed to memory prior to this synchronization point completing. This bit must be set for all write fence sync operations to assure that results from operations initiated prior to this command are visible in memory once software observes this synchronization.

Value	Name	Description
0		Tile Cache is not flushed.
1		Tile cache is flushed.

Programming Notes

- SW must always set CS Stall bit when Tile Cache Flush Enable bit is set in the PIPECONTROL command.
- SW must ensure level1 depth and color caches are flushed prior to flushing the tile cache. This can be achieved by following means:
 - Single PIPECONTROL command to flush level1 caches and the tile cache. Attributes listed below must be set. OR

PIPE_CONTROL

		<ul style="list-style-type: none"> • Tile Cache Flush Enable • Render Target Cache Flush Enable • Depth Cache Flush Enable • Flushing of L1 caches followed by flushing of tile cache through two different PIPECONTROL commands. SW must ensure not to issue any rendering commands between the two PIPECONTROL commands. 				
		<p>Tile Cache Enabled Mode:</p> <ul style="list-style-type: none"> • SW must always set CS Stall bit when Tile Cache Flush Enable bit is set in the PIPECONTROL command. • SW must ensure level1 depth and color caches are flushed prior to flushing the tile cache. This can be achieved by following means: <ul style="list-style-type: none"> • Single PIPECONTROL command to flush level1 caches and the tile cache. Hardware will sequence the flushing of L1 caches followed by the Tile cache. Attributes listed below must be set. OR • Tile Cache Flush Enable • Render Target Cache Flush Enable • Depth Cache Flush Enable • Flushing of L1 caches followed by flushing of tile cache through two different PIPECONTROL commands. SW must ensure not to issue any rendering commands between the two PIPECONTROL commands. <p>Unified Cache (Tile Cache Disabled):</p> <p>In unified cache mode of operation Color and Depth (Z) streams are cached in DC space of L2 along with Data Port stream. On a Tile Cache Flush only Color and Depth (Z) streams from DC space of L2 are flushed to globally observable and where as DC Flush Enable will only flush Data Port stream from the DC space of L2 to globally observable. Refer L3 configuration section for Unified cache usage model. In this mode of operation there is no dedicated memory allocated for Tile Cache in L2. When the Color and Depth (Z) streams are enabled to be cached in the DC space of L2, Software must use Render Target Cache Flush Enable and Depth Cache Flush Enable along with Tile Cache Flush for getting the color and depth (Z) write data to be globally observable. In this mode of operation it is not required to set CS Stall upon setting Tile Cache Flush bit.</p>				
27	Reserved					
26	Flush LLC	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">Enable</td> </tr> </table> <p>If enabled, at the end of the current pipe-control the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; color: blue; font-weight: bold;">Programming Notes</td> </tr> <tr> <td>SW must always program Post-Sync Operation to "Write Immediate Data" when Flush LLC is set.</td> </tr> </table>	Format:	Enable	Programming Notes	SW must always program Post-Sync Operation to "Write Immediate Data" when Flush LLC is set.
Format:	Enable					
Programming Notes						
SW must always program Post-Sync Operation to "Write Immediate Data" when Flush LLC is set.						
25	AMFS Flush Enable					

PIPE_CONTROL											
	Format:	Enable									
	<p>If enabled, at the end of the current pipe-control the AMFS unit stalls until all spawned texel shaders are completed, and then the AMFS unit flushes internal cache(s) to memory. This bit should be enabled when a procedural texture transitions from the write state to the read state.</p>										
24	<p>Destination Address Type Defines address space of Destination Address</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>PPGTT</td> <td>Use PPGTT address space for DW write</td> </tr> <tr> <td>1h</td> <td>GGTT</td> <td>Use GGTT address space for DW write</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Ignored if ""No Write" is selected in Operation.</p>		Value	Name	Description	0h	PPGTT	Use PPGTT address space for DW write	1h	GGTT	Use GGTT address space for DW write
Value	Name	Description									
0h	PPGTT	Use PPGTT address space for DW write									
1h	GGTT	Use GGTT address space for DW write									
23	<p>LRI Post Sync Operation</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>No LRI Operation</td> <td>No LRI operation occurs as a result of this instruction. The Post-Sync Operation field is valid and may be used to specify an operation.</td> </tr> <tr> <td>1h</td> <td>MMIO Write Immediate Data</td> <td>Write the DWord contained in Immediate Data Low (DW3) to the MMIO offset specified in the Address field.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>This bit causes a post sync operation with an LRI (Load Register Immediate) operation. If this bit is set then the Post-Sync Operation field must be cleared.</p>		Value	Name	Description	0h	No LRI Operation	No LRI operation occurs as a result of this instruction. The Post-Sync Operation field is valid and may be used to specify an operation.	1h	MMIO Write Immediate Data	Write the DWord contained in Immediate Data Low (DW3) to the MMIO offset specified in the Address field.
Value	Name	Description									
0h	No LRI Operation	No LRI operation occurs as a result of this instruction. The Post-Sync Operation field is valid and may be used to specify an operation.									
1h	MMIO Write Immediate Data	Write the DWord contained in Immediate Data Low (DW3) to the MMIO offset specified in the Address field.									
22	Reserved										
21	<p>Store Data Index</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).</p>		Format:	U1							
Format:	U1										
20	<p>Command Streamer Stall Enable</p> <table border="1"> <tr> <td>Format:</td> <td>U1</td> </tr> </table> <p>If ENABLED, the sync operation will not occur until all previous flush operations pending a completion of those previous flushes will complete, including the flush produced from this command. This enables the command to act similar to the legacy MI_FLUSH command.</p>		Format:	U1							
Format:	U1										
19	Reserved										
18	TLB Invalidate										

PIPE_CONTROL

		Format:	U1
		<p>If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting</p> <p>If ENABLED, PIPE_CONTROL command will flush the in flight data written out by render engine to Global Observation point on flush done. Also Requires stall bit ([20] of DW1) set.</p>	
		Programming Notes	
		<p>If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting.</p> <p>Post Sync Operation or CS stall must be set to ensure a TLB invalidate occurs. Otherwise no cycle will occur to the TLB cache to invalidate.</p>	
17	PSD Sync Enable	Format:	Enable
		<p>If set, Pixel Shader Dispatch Units will stall successive PS threads from being dispatched until all the prior PS threads complete. Once all PSDs are synced up, post-sync LRI can be optionally used to change EU enables.</p>	
		Programming Notes	
		<p>If this bit is set, these bits must not be set:</p> <ul style="list-style-type: none"> • Depth Stall Enable • Command Streamer Stall Enable • Render Target Cache Flush Enable • Stall At Pixel Scoreboard 	
16	Generic Media State Clear	Format:	Disable
		<p>If set, all generic media state context information will be invalidated. Any state invalidated will not be saved as part of the render engine context image. The state only becomes valid once it is parsed by the command streamer.</p>	
15:14	Post Sync Operation	Description	
		<p>This field specifies an optional action to be taken upon completion of the synchronization operation.</p> <p>This field must be cleared if the LRI Post-Sync Operation bit is set.</p>	
		Value	Name
		Description	Programming Notes
		0h	No Write
		No write occurs as a result of this instruction. This can be used to	

PIPE_CONTROL			
		implement a "trap" operation, etc.	
1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address	
2h	Write PS Depth Count	Write the 64-bit PS_DEPTH_COUNT register to the Destination Address	[] Workaround : Driver must program PIPE_CONTROL with only Depth Stall Enable bit set prior to programming a PIPE_CONTROL with Write PS Depth Count Post sync operation.
3h	Write Timestamp	Write the 64-bit TIMESTAMP register(i.e. "Reported Timestamp Count" 0x2358 for render pipe) to the Destination Address.	
Programming Notes			
If executed in non-secure batch buffer, the address given will be in a PPGTT address space. If in a secure ring or batch, address given will be in GGTT space			
13	Depth Stall Enable		
Format:		Enable	
This bit must be set when obtaining a "visible pixel" count to preclude the possible inclusion in the PS_DEPTH_COUNT value written to memory of some fraction of pixels from objects initiated after the PIPE_CONTROL command.			
Value	Name	Description	
0h	Disable	3D pipeline will not stall subsequent primitives at the Depth Test stage.	
1h	Enable	3D pipeline will stall any subsequent primitives at the Depth Test stage until the Sync and Post-Sync operations complete.	
Programming Notes			
This bit must be DISABLED for operations other than writing PS_DEPTH_COUNT.			
This bit will have no effect (besides preventing write cache flush) if set in a PIPE_CONTROL command issued to the Media pipe.			
12	Render Target Cache Flush Enable		
Format:		Enable	
Setting this bit will force Render Cache to be flushed to memory prior to this synchronization point completing. This bit must be set for all write fence sync operations to assure that results from operations initiated prior to this command are visible in memory once software observes this synchronization.			
Value	Name	Description	
0h	Disable Flush	Render Target Cache is NOT flushed.	
1h	Enable Flush	Render Target Cache is flushed.	

PIPE_CONTROL													
	<table border="1" style="width: 100%;"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">This bit must not be set when Depth Stall Enable bit is set in this packet.</td> </tr> <tr> <td colspan="2">Whenever a Binding Table Index (BTI) used by a Render Target Message points to a different RENDER_SURFACE_STATE, SW must issue a Render Target Cache Flush by enabling this bit.</td> </tr> <tr> <td colspan="2">When render target flush is set due to new association of BTI, PS Scoreboard Stall bit must be set in this packet.</td> </tr> </tbody> </table>	Programming Notes		This bit must not be set when Depth Stall Enable bit is set in this packet.		Whenever a Binding Table Index (BTI) used by a Render Target Message points to a different RENDER_SURFACE_STATE, SW must issue a Render Target Cache Flush by enabling this bit.		When render target flush is set due to new association of BTI, PS Scoreboard Stall bit must be set in this packet.					
Programming Notes													
This bit must not be set when Depth Stall Enable bit is set in this packet.													
Whenever a Binding Table Index (BTI) used by a Render Target Message points to a different RENDER_SURFACE_STATE, SW must issue a Render Target Cache Flush by enabling this bit.													
When render target flush is set due to new association of BTI, PS Scoreboard Stall bit must be set in this packet.													
11	<p>Instruction Cache Invalidate Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 at the top of the pipe i.e. at the parsing time.</p>	Format:	Enable										
Format:	Enable												
10	<p>Texture Cache Invalidation Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the texture caches at the top of the pipe i.e. at the parsing time.</p>	Format:	Enable										
Format:	Enable												
9	<p>Indirect State Pointers Disable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <table border="1" style="width: 100%;"> <thead> <tr> <th colspan="2" style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td colspan="2">At the completion of the post-sync operation associated with this pipe control packet, the indirect state pointers in the hardware are considered invalid; the indirect pointers are not saved in the context. If any new indirect state commands are executed in the command stream while the pipe control is pending, the new indirect state commands are preserved.</td> </tr> <tr> <td colspan="2">Using Invalidate State Pointer (ISP) only inhibits context restoring of Push Constant (3DSTATE_CONSTANT_*) commands. Push Constant commands are only considered as Indirect State Pointers. Once ISP is issued in a context, SW must initialize by programming push constant commands for all the shaders (at least to zero length) before attempting any rendering operation for the same context.</td> </tr> </tbody> </table> <table border="1" style="width: 100%;"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">When executed in POCS results in inhibiting the context restore of 3DSTATE_CONSTANT_VS from POSH engine context.</td> </tr> </tbody> </table>	Format:	Enable	Description		At the completion of the post-sync operation associated with this pipe control packet, the indirect state pointers in the hardware are considered invalid; the indirect pointers are not saved in the context. If any new indirect state commands are executed in the command stream while the pipe control is pending, the new indirect state commands are preserved.		Using Invalidate State Pointer (ISP) only inhibits context restoring of Push Constant (3DSTATE_CONSTANT_*) commands. Push Constant commands are only considered as Indirect State Pointers. Once ISP is issued in a context, SW must initialize by programming push constant commands for all the shaders (at least to zero length) before attempting any rendering operation for the same context.		Programming Notes		When executed in POCS results in inhibiting the context restore of 3DSTATE_CONSTANT_VS from POSH engine context.	
Format:	Enable												
Description													
At the completion of the post-sync operation associated with this pipe control packet, the indirect state pointers in the hardware are considered invalid; the indirect pointers are not saved in the context. If any new indirect state commands are executed in the command stream while the pipe control is pending, the new indirect state commands are preserved.													
Using Invalidate State Pointer (ISP) only inhibits context restoring of Push Constant (3DSTATE_CONSTANT_*) commands. Push Constant commands are only considered as Indirect State Pointers. Once ISP is issued in a context, SW must initialize by programming push constant commands for all the shaders (at least to zero length) before attempting any rendering operation for the same context.													
Programming Notes													
When executed in POCS results in inhibiting the context restore of 3DSTATE_CONSTANT_VS from POSH engine context.													
8	<p>Notify Enable</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p>If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.</p>	Format:	Enable										
Format:	Enable												
7	<p>Pipe Control Flush Enable</p>												

PIPE_CONTROL			
	Format:	Enable	
	Hardware on parsing PIPECONTROL command with Pipe Control Flush Enable set will wait for all the outstanding post sync operations corresponding to previously executed PIPECONTROL commands are complete before making forward progress.		
6	Reserved		
5	DC Flush Enable		
	Format:	Enable	
	Setting this bit enables flushing of the L3\$ portions that caches DC writes.		
	Programming Notes		
	DC Flush (L3 Flush) by default doesn't result in flushing/invalidating the IA Coherent lines from L3\$, however this can be achieved by setting control bit Pipe line flush Coherent lines in L3SQCREG4 register.		
4	VF Cache Invalidation Enable		
	Format:	Enable	
	Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of VF address based cache at the top of the pipe i.e. at the parsing time.		
	Programming Notes		
	When executed from POCS it results in invalidating the VFR cache.		
3	Constant Cache Invalidation Enable		
	Format:	Enable	
	Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the constant cache at the top of the pipe i.e. at the parsing time.		
2	State Cache Invalidation Enable		
	Format:	Enable	
	Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 state caches at the top of the pipe i.e. at the parsing time.		
	Programming Notes		
	When State Cache redirect to CS Section enable bit is set in MMIO register SLICE_COMMON_ECO_CHICKEN1 (0731Ch), Command Cache Invalidate Enable must be also set upon setting State Cache Invalidate bit in PIPE_CONTROL command.		
1	Stall At Pixel Scoreboard		
	Format:	Enable	
	Defines the behavior of PIPE_CONTROL command at the pixel scoreboard.		
	Value	Name	
	Description		
	0h	Disable	Stall at the pixel scoreboard is disabled.
	1h	Enable	Stall at the pixel scoreboard is enabled.
	Programming Notes		

PIPE_CONTROL													
		This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries. This bit is ignored if Depth Stall Enable is set. Further the render cache is not flushed even if Write Cache Flush Enable bit is set.											
	0	<p>Depth Cache Flush Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Setting this bit enables flushing (i.e. writing back the dirty lines to memory and invalidating the tags) of depth related caches. This bit applies to HiZ cache, Stencil cache and depth cache.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Flush Disabled</td> <td>Depth relates caches (HiZ, Stencil and Depth) are NOT flushed.</td> </tr> <tr> <td>1h</td> <td>Flush Enabled</td> <td>Depth relates caches (HiZ, Stencil and Depth) are flushed.</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Ideally depth caches need to be flushed only when depth is required to be coherent in memory for later use as a texture, source or honoring CPU lock. This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries.</p>	Format:	Enable	Value	Name	Description	0h	Flush Disabled	Depth relates caches (HiZ, Stencil and Depth) are NOT flushed.	1h	Flush Enabled	Depth relates caches (HiZ, Stencil and Depth) are flushed.
Format:	Enable												
Value	Name	Description											
0h	Flush Disabled	Depth relates caches (HiZ, Stencil and Depth) are NOT flushed.											
1h	Flush Enabled	Depth relates caches (HiZ, Stencil and Depth) are flushed.											
2	31:2	<p>Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[31:2]U32</td> </tr> </table> <p>If Post Sync Operation is set to 1h LRI Post-Sync Operation must be clear): Bits 31:3 specify the QW address of where the Immediate Data following this DW in the packet to be stored. Bit 2 MBZ Ignored if "No Write" is the selected in Post-Sync Operation If LRI Post-Sync Operation is set: Bits 22:2 (Bits 31:23 are reserved MBZ) specify the MMIO offset destination for the data in the Immediate Data Low (DW3) field. Only DW writes are valid.</p>	Format:	GraphicsAddress[31:2]U32									
Format:	GraphicsAddress[31:2]U32												
	1:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
3	31:0	<p>Address High</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[63:32]U32</td> </tr> </table> <p>This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space. This field is valid only if the post-sync operation is not 0 and the LRI Post-Sync Operation is clear.</p>	Format:	GraphicsAddress[63:32]U32									
Format:	GraphicsAddress[63:32]U32												
4..5	63:0	<p>Immediate Data</p> <table border="1"> <tr> <td>Format:</td> <td>U64</td> </tr> </table> <p>This field specifies the QWord value to be written to the targeted location. Only valid when Post-Sync Operation is 1h (Write Immediate Data) or LRI Post-Sync Operation is set. Ignored if Post-Sync Operation is "No write", "Write PS_DEPTH_COUNT" or "Write TIMESTAMP".</p>	Format:	U64									
Format:	U64												

PIPELINE_SELECT

PIPELINE_SELECT	
Source:	BSpec
Length Bias:	1
Description	
The PIPELINE_SELECT command is used to specify which GPE pipeline is to be considered the 'current' active pipeline.	
Issuing 3D-pipeline-specific commands when the GPGPU pipeline is selected, or vice versa, is UNDEFINED.	
Programming common non pipeline commands (e.g., STATE_BASE_ADDRESS) is allowed in all pipeline modes.	
Programming Notes	
<p>Software must ensure Render Cache, Depth Cache and HDC Pipeline flush are flushed through a stalling PIPE_CONTROL command prior to programming of PIPELINE_SELECT command transitioning Pipeline Select from 3D to GPGPU/Media.</p> <p>Software must ensure HDC Pipeline flush and Generic Media State Clear is issued through a stalling PIPE_CONTROL command prior to programming of PIPELINE_SELECT command transitioning Pipeline Select from GPGPU/Media to 3D.</p> <p>Example:</p> <ul style="list-style-type: none"> • Workload-3Dmode, • PIPE_CONTROL (CS Stall, Depth Cache Flush Enable, Render Target Cache Flush Enable, HDC Pipeline Flush Enable) , • PIPELINE_SELECT (GPGPU), • Workload-GPGPUmode, • PIPE_CONTROL (CS Stall, HDC Pipeline Flush Enable, Generic Media State Clear), • PIPELINE_SELECT (3D) ... 	
"Pipe Selection" must be never set to "3D" in PIPELINE_SELECT command programmed for workloads submitted to ComputeCS.	
<p>While GPU is operating in GPGPU mode of operation and when a Mid Thread Preemption (if enabled) occurs on a PIPELINE_SELECT command with Media Sampler DOP CG Enable reset along with Pipeline Select Mode set to 3D and on resubmission of this context on context restore Sampler DOP CG Enable will be reset. This would mean the GPGPU mid thread preempted threads restored will get executed with media sampler DOP clock not gated consuming media sampler DOP power until all GPGPU threads have retired.</p> <p>Programming of the PIPELINE_SELECT can be modified to avoid the above inefficiency. This can be done by programming Pipeline Selection and Media Sampler DOP CG Enable fields in two different PIPELINE_SELECT commands instead of on single PIPELINE_SELECT command.</p> <p>Example:</p> <p>PIPELINE_SELECT (Pipeline Selection = 3D, Media Sampler DOP CG Enable = False)</p> <p>To</p> <p>PIPELINE_SELECT (Pipeline Selection = 3D)</p> <p>PIPELINE_SELECT (Media Sampler DOP CG Enable = False)</p>	

DWord	Bit	Description								
0	31:29	Command Type								
		Default Value: 3h GFXPIPE Format: OpCode								
	28:27	Command SubType								
		Default Value: 1h GFXPIPE_SINGLE_DW Format: OpCode								
	26:24	3D Command Opcode								
		Default Value: 1h GFXPIPE_NONPIPELINED Format: OpCode								
	23:16	3D Command Sub Opcode								
		Default Value: 04h PIPELINE_SELECT Format: OpCode								
	15:8	Mask Bits								
		Programming Notes Must be set to modify corresponding bits in Bits 7:0. (For implemented bits)								
	7	Reserved								
		Format: MBZ								
6	Media Sampler Power Clock Gate Disable									
	Format: U1									
	By default, the media power clock gating is always ON. When set, Command Streamer sends message to PM to disable media sampler power Clock Gating.									
	Programming Notes Mask bit [14] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed. Each BB/workload is responsible to set this control. This can be only enabled/disabled at the frame level.									
5	Reserved									
	Format: MBZ									
4	Media Sampler DOP Clock Gate Enable									
	Format: Enable									
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disabled</td> <td>Command Streamer sends message to PM to disable sampler DOP Clock Gating.</td> </tr> <tr> <td>1</td> <td>Enabled</td> <td>Command Streamer sends message to PM to enable media sampler DOP Clock Gating.</td> </tr> </tbody> </table>	Value	Name	Description	0	Disabled	Command Streamer sends message to PM to disable sampler DOP Clock Gating.	1	Enabled	Command Streamer sends message to PM to enable media sampler DOP Clock Gating.
	Value	Name	Description							
0	Disabled	Command Streamer sends message to PM to disable sampler DOP Clock Gating.								
1	Enabled	Command Streamer sends message to PM to enable media sampler DOP Clock Gating.								

PIPELINE_SELECT

Programming Notes		
Mask bit [12] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.		
3	Render Sampler Power Gate Enable Format: Enable	
Value	Name	Description
0	Disabled	Command Streamer sends message to PM to disable render sampler Power Gating.
1	Enabled	Command Streamer sends message to PM to enable render sampler Power Gating.
Programming Notes		
Mask bit [11] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.		
2	Render Slice common Power Gate Enable Format: Enable	
Value	Name	Description
0	Disabled	Command Streamer sends message to PM to disable render slice common Power Gating.
1	Enabled	Command Streamer sends message to PM to enable render slice common Power Gating.
Programming Notes		
Mask bit [10] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.		
1:0	Pipeline Selection	
Value	Name	Description
0	3D	3D pipeline is selected
1	Media	Media pipeline is selected (Includes HD optical disc playback, HD video playback, and generic media workloads)
2	GPGPU	GPGPU pipeline is selected
Programming Notes		
Mask bits [9:8] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed. Setting only one of the mask bit [9] or [8] is illegal.		

Read Surface Info MSD

MSD_RSI - Read Surface Info MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHF Indicates that the message forbids a header.
	18	Reserved Format: MBZ
	17:14	Message Type Default Value: 06h Format: Opcode Read Surface Info message
	13:8	Reserved Format: MBZ
	7:0	Binding Table Index Format: MDC_BTS Specifies the Binding Table Index for the message

REP16 Render Target Write MSD

MSD_RTW_REP16 - REP16 Render Target Write MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 0h Format: Opcode Full precision data message
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. Programming Notes This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	Message Type Default Value: 0Ch

MSD_RTW_REP16 - REP16 Render Target Write MSD

		Format:	Opcode
		Render Target Write message	
13	Per-Sample PS Enable	Format:	Enable
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	<p>This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.</p> <p>When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.</p> <p>When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).</p> <p>When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.</p>		
12	Last Render Target Select	Format:	Enable
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select	Format:	MDC_RT_SGS
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype	Default Value:	1h
		Format:	Opcode
	SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].		
7:0	Binding Table Index	Format:	MDC_BTS
	Specifies the Binding Table Index for the message		

Reserved Instruction0

Reserved Instruction0		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:24	Opcode 2 Format: Opcode
	23:16	Opcode 3 Format: Opcode
	15:8	Reserved Format: MBZ
	7:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield



Reserved Instruction1

Reserved Instruction1				
Length Bias:	1			
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1"><tr><td></td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:23	Opcode 1 Format: <table border="1"><tr><td></td><td>Opcode</td></tr></table>		Opcode
	Opcode			
22:0	Reserved Format: <table border="1"><tr><td></td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
0..n	31:0	Unknown Bitfield		

Reserved Instruction2

Reserved Instruction2				
Length Bias: 1				
DWord	Bit	Description		
0	31:29	Opcode 0 <table border="1"> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
	28:23	Opcode 1 <table border="1"> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
Format:	Opcode			
22:0	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
0..n	31:0	Unknown Bitfield		



Reserved Instruction3

Reserved Instruction3				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:16	Opcode 1 Format: <table border="1"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
	Opcode			
15:0	Reserved Format: <table border="1"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
0..n	31:0	Unknown Bitfield		

Reserved Instruction4

Reserved Instruction4			
Length Bias: 2			
DWord	Bit	Description	
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
	28:16	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
15:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> =n		
0..n	31:0	Unknown Bitfield	



Reserved Instruction5

Reserved Instruction5				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:16	Opcode 1 Format: <table border="1"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
	Opcode			
15:0	DWord Count Format: <table border="1"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		

Reserved Instruction6

Reserved Instruction6				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 <table border="1"> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
	28:16	Opcode 1 <table border="1"> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
Format:	Opcode			
15:0	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
0..n	31:0	Unknown Bitfield		



Reserved Instruction7

Reserved Instruction7				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	26:24	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	23:21	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
	Opcode			
20:16	Opcode 4 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode	
	Opcode			
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		

Reserved Instruction8

Reserved Instruction8		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield



Reserved Instruction9

Reserved Instruction9		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield

Reserved Instruction10

Reserved Instruction10		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield



Reserved Instruction11

Reserved Instruction11		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield

Reserved Instruction12

Reserved Instruction12		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield



Reserved Instruction13

Reserved Instruction13				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	26:23	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
22:16	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode	
	Opcode			
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		

Reserved Instruction14

Reserved Instruction14				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	26:23	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
22:16	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode	
	Opcode			
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		



Reserved Instruction15

Reserved Instruction15		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield

Reserved Instruction16

Reserved Instruction16				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
	28:27	Opcode 1 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
	26:23	Opcode 2 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
22:16	Opcode 3 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode	
Format:	Opcode			
15:12	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
11:0	DWord Count <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>=n</td> </tr> </table>	Format:	=n	
Format:	=n			
0..n	31:0	Unknown Bitfield		



Reserved Instruction17

Reserved Instruction17		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield

Reserved Instruction18

Reserved Instruction18		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield



Reserved Instruction19

Reserved Instruction19				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	26:23	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
22:16	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode	
	Opcode			
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		

Reserved Instruction20

Reserved Instruction20		
Length Bias: 2		
DWord	Bit	Description
0	31:29	Opcode 0 Format: Opcode
	28:27	Opcode 1 Format: Opcode
	26:23	Opcode 2 Format: Opcode
	22:16	Opcode 3 Format: Opcode
	15:12	Reserved Format: MBZ
	11:0	DWord Count Format: =n
0..n	31:0	Unknown Bitfield



Reserved Instruction21

Reserved Instruction21				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	26:23	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
22:16	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode	
	Opcode			
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		

Reserved Instruction22

Reserved Instruction22				
Length Bias:		2		
DWord	Bit	Description		
0	31:29	Opcode 0 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
	28:27	Opcode 1 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
	26:23	Opcode 2 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
	Format:	Opcode		
	22:21	Opcode 3 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode
Format:	Opcode			
20:16	Opcode 4 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>Opcode</td> </tr> </table>	Format:	Opcode	
Format:	Opcode			
15:12	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
11:0	DWord Count <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>=n</td> </tr> </table>	Format:	=n	
Format:	=n			
0..n	31:0	Unknown Bitfield		



Reserved Instruction23

Reserved Instruction23			
Length Bias: 2			
DWord	Bit	Description	
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
	26:23	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
	22:21	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
20:16	Opcode 4 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode		
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> MBZ		
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> =n		
0..n	31:0	Unknown Bitfield	

Reserved Instruction24

Reserved Instruction24				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	26:23	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	22:21	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
	Opcode			
20:16	Opcode 4 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode	
	Opcode			
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		



Reserved Instruction25

Reserved Instruction25			
Length Bias: 2			
DWord	Bit	Description	
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
	26:23	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
	22:21	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode	
20:16	Opcode 4 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> Opcode		
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> MBZ		
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td></tr></table> =n		
0..n	31:0	Unknown Bitfield	

Reserved Instruction26

Reserved Instruction26				
Length Bias: 2				
DWord	Bit	Description		
0	31:29	Opcode 0 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	28:27	Opcode 1 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	26:24	Opcode 2 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
		Opcode		
	23:21	Opcode 3 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode
	Opcode			
20:16	Opcode 4 Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>Opcode</td></tr></table>		Opcode	
	Opcode			
15:12	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>MBZ</td></tr></table>		MBZ	
	MBZ			
11:0	DWord Count Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td> </td><td>=n</td></tr></table>		=n	
	=n			
0..n	31:0	Unknown Bitfield		

Return

ret - Return			
Source:	Eulsa		
Length Bias:	4		
<p>Return execution to the code sequence that called a subroutine. The ret instruction can be predicated or non-predicated. If non-predicated, all channels jump to the return IP in the first channel of src0 and restore CallMask from the second channel of src0. If predicated, the enabled channels jump to the return IP from the first channel of src0 and the corresponding bits in the CallMask are cleared to zero; if all CallMask bits are zero after the ret instruction, then execution jumps to the return IP from the first channel of src0. When SPF is on, the predication control must be scalar.</p>			
Format:	[(pred)] ret (exec_size) null src0		
Restriction			
This instruction cannot take accumulator as source.			
Syntax			
[(pred)] ret (exec_size) null reg			
Pseudocode			
<pre>Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { PcIP[n] = src0.chan[0]; CallMask[n] = 0; } else { PcIP[n] = IP + 1; } } for (n = exec_size; n < 32; n++) { PcIP[n] = IP + 1; } if (CallMask[n:0] == 0) { // all channels are zero Jump(src0.chan[0]); CallMask = src0.chan[1]; }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0.3	127:96	Reserved	
		Exists If:	([Src0.IsImm]==false)
	Format:	MBZ	
	127:96	JIP	
		Exists If:	([Src0.IsImm]==true)
		Format:	S31
The byte-aligned jump distance if a jump is taken for the channel			
95:80	Reserved		
	Exists If:	([Src0.IsImm]==false)	
Format:	MBZ		

ret - Return

95:64	Reserved	
	Exists If:	([Src0.IsImm]==true)
	Format:	MBZ
79:66	Src0.Operand	
	Exists If:	([Src0.IsImm]==false)
	Format:	DirectOperand
65:64	Reserved	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
63:50	Dst.Operand	
	Format:	DirectOperand
49:47	Reserved	
	Format:	MBZ
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false
	1	true
45:34	Reserved	
	Format:	MBZ
33	BranchCtrl	
	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
	0	Normal [Default]
	1	NoMask
		Description
		Normal. Per channel write enable used for final write enable generation.
		NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU	

ret - Return

	<p>decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.								
28	<p>PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									

ret - Return			
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Rotate Left

rol - Rotate Left			
Source:	Eulsa		
Length Bias:	4		
<p>Perform component-wise logical rotate left operation of the bits in src0 by the rotate count indicated in src1, storing the result in dst. src0 and src1 are treated as unsigned numbers with only the bits within the specified datatype used during this operation. This operation does not produce sign or overflow conditions. Only the .e/z or .ne/.nz conditional modifiers are supported. Extra precision bits available in accumulator are ignored during this operation and only the bits within the specified datatype are used. src0 and dst must be of same datatype precision.</p>			
Format:	[(pred)] rol[.cmod] (exec_size) dst src0 src1		
Syntax			
<pre>[(pred)] rol[.cmod] (exec_size) reg reg reg [(pred)] rol[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { RotateMask = sizeof(src0) * 8 - 1; dst.chan[n] = (src0.chan[n] << (src1.chan[n] & RotateMask)) (src0.chan[n] >> (-src1.chan[n] & RotateMask)); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N
Src Types	Dst Types		
UW, UD	UW, UD		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
	125:122	Reserved	
Exists If:		([Src1.IsImm]==false)	
Format:		MBZ	

rol - Rotate Left

121:120	Src1.Mod	
	Exists If:	([Src1.IsImm]==false)
	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	([Src1.IsImm]==false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	([Src1.IsImm]==false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	

rol - Rotate Left

	Exists If:	([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Src1.IsImm		
	This field indicate that Source 1 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==false)	
	Format:	RegDataType	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==true)	
	Format:	ImmDataType	
39:36	Dst.DataType		
	Format:	RegDataType	
35	Dst.AddrMode		

rol - Rotate Left

	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
		Description
	0	Normal [Default]
		Normal. Per channel write enable used for final write enable generation.
	1	NoMask
		NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name
		Description
	0	NoCompaction [Default]
		No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted
		Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv	
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name
		Description
	0	Positive [Default]
		Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative
		Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

rol - Rotate Left

27:24	<p>PredCtrl</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl
Format:	PredCtrl		
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Rotate Right

ror - Rotate Right			
Source:	Eulsa		
Length Bias:	4		
<p>Perform component-wise logical rotate right operation of the bits in src0 by the rotate count indicated in src1, storing the result in dst. src0 and src1 are treated as unsigned numbers with only the bits within the specified datatype used during this operation. This operation does not produce sign or overflow conditions. Only the .e/z or .ne/.nz conditional modifiers are supported. Extra precision bits available in accumulator are ignored during this operation and only the bits within the specified datatype are used. src0 and dst must be of same datatype precision.</p>			
Format:	[(pred)] ror[.cmod] (exec_size) dst src0 src1		
Syntax			
<pre>[(pred)] ror[.cmod] (exec_size) reg reg reg [(pred)] ror[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { RotateMask = sizeof(src0) * 8 - 1; dst.chan[n] = (src0.chan[n] » (src1.chan[n] & RotateMask)) (src0.chan[n] « (-src1.chan[n] & RotateMask)); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	N	N
Src Types	Dst Types		
UW, UD	UW, UD		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
	125:122	Reserved	
Exists If:		([Src1.IsImm]==false)	
Format:		MBZ	

ror - Rotate Right

121:120	Src1.Mod	
	Exists If:	([Src1.IsImm]==false)
	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	([Src1.IsImm]==false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	([Src1.IsImm]==false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	([Src1.IsImm]==false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	([Src1.IsImm]==true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	

ror - Rotate Right		
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	

ror - Rotate Right

		Format:	AddrMode									
34	Saturate	Format: Saturate										
33	AccWrCtrl	Format: AccWrCtrl										
32	AtomicCtrl	Format: AtomicCtrl										
31	MaskCtrl	<p>Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td>1</td> <td>NoMask</td> <td>NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>		Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description										
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.										
1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.										
30	Reserved											
29	CmptCtrl	Format:	MBZ									
		<p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>		Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description										
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.										
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.										
28	PredInv	<p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 75%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										

ror - Rotate Right

27:24	<p>PredCtrl</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl
Format:	PredCtrl		
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Round Down

rndd - Round Down			
Source:	Eulsa		
Length Bias:	4		
<p>The rndd instruction takes component-wise floating point downward rounding (to the integral float number closer to negative infinity) of src0 and storing the rounded integral float results in dst. This is commonly referred to as the floor() function. Each result follows the rules in the following tables based on the floating-point mode.</p>			
<p>Format:</p> <pre>[(pred)] rndd[.cmod] (exec_size) dst src0</pre>			
Syntax			
<pre>[(pred)] rndd[.cmod] (exec_size) reg reg [(pred)] rndd[.cmod] (exec_size) reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = floor(src0.chan[n]); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
F	F		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0] Exists If: ([Src0.IsImm] == true)	
	95:92	CondCtrl Exists If: ([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df)) Format: FlagModifier	
	95:64	Src0.ImmValue[63:32] Exists If: ([Src0.IsImm] == true) AND (([Src0.DataType] == :q) OR ([Src0.DataType] == :uq) OR ([Src0.DataType] == :df))	
	87:84	Src0.VertStride Exists If: ([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND ([Src0.DataType] != :df)) Format: VertStride	
	83:81	Src0.Width Exists ([Src0.IsImm] == false) OR (([Src0.DataType] != :q) AND ([Src0.DataType] != :uq) AND	

rndd - Round Down

		If:	([Src0.DataType]!=:df)	
		Format:	Width	
80	Src0.AddrMode			
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))		
	Format:	AddrMode		
79:66	Src0.Operand			
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)		
	Format:	DirectOperand		
79:66	Src0.Operand			
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)		
	Format:	IndirectOperand		
65:64	Src0.HorzStride			
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))		
	Format:	HorzStride		
63:50	Dst.Operand			
	Exists If:	([Dst.AddrMode]==Indirect)		
	Format:	IndirectOperand		
63:50	Dst.Operand			
	Exists If:	([Dst.AddrMode]==Direct)		
	Format:	DirectOperand		
49:48	Dst.HorzStride			
	Format:	HorzStride		
47	Reserved			
	Format:	MBZ		
46	Src0.IsImm			
	This field indicate that Source 0 operand is carrying an immediate value.			
	Value	Name		
	0	false [Default]		
	1	true		
45:44	Src0.Mod			
	Format:	SrcMod		
43:40	Src0.DataType			
	Exists If:	([Src0.IsImm]==false)		

rndd - Round Down

		Format:	RegDataType
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm] == true)	
	Format:	ImmDataType	
39:36	Dst.DataType		
	Format:	RegDataType	
35	Dst.AddrMode		
	Format:	AddrMode	
34	Saturate		
	Format:	Saturate	
33	AccWrCtrl		
	Format:	AccWrCtrl	
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
30	Reserved		
29	CmptCtrl		
	Format:	MBZ	
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.	
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits		

rndd - Round Down

	<p>generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		Format:	PredCtrl							
Format:	PredCtrl										
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>										
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>										
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>		Format:	ChanOff							
Format:	ChanOff										
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>		Format:	ExecSize							
Format:	ExecSize										
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>		Format:	Header							
Format:	Header										

Round to Nearest or Even

rnde - Round to Nearest or Even			
Source:	Eulsa		
Length Bias:	4		
<p>The rnde instruction takes component-wise floating point round-to-even operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-even increments stored in the rounding increment bits. The round-to-even increment must be added to the results in dst to create the final round-to-even values to emulate the round-to-even operation, commonly known as the round() function. The final results are the one of the two integral float values that is nearer to the input values. If the neither possibility is nearer, the even alternative is chosen. Each result follows the rules in the following tables based on the floating-point mode.</p>			
Format:	<code>[(pred)] rnde[.cmod] (exec_size) dst src0</code>		
Syntax			
<code>[(pred)] rnde[.cmod] (exec_size) reg reg [(pred)] rnde[.cmod] (exec_size) reg imm32</code>			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { if (src0.chan[n] - floor(src0.chan[n]) > 0.5f) { dst.chan[n] = floor(src0.chan[n]) + 1; } else if (src0.chan[n] - floor(src0.chan[n]) < 0.5f) { dst.chan[n] = floor(src0.chan[n]); } else { if (floor(src0.chan[n]) is odd) { dst.chan[n] = floor(src0.chan[n]) + 1; } else { dst.chan[n] = floor(src0.chan[n]); } } } } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
F	F		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0] Exists If: <code>[(Src0.IsImm)]==true</code>	
	95:92	CondCtrl Exists If: <code>[(Src0.IsImm)]==false</code> OR <code>(([(Src0.DataType)]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))</code> Format: FlagModifier	
	95:64	Src0.ImmValue[63:32] Exists If: <code>[(Src0.IsImm)]==true</code> AND <code>(([(Src0.DataType)]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))</code>	

rnde - Round to Nearest or Even

87:84	Src0.VertStride	
Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
Format:	VertStride	
83:81	Src0.Width	
Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
Format:	Width	
80	Src0.AddrMode	
Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
Format:	AddrMode	
79:66	Src0.Operand	
Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)	
Format:	DirectOperand	
79:66	Src0.Operand	
Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)	
Format:	IndirectOperand	
65:64	Src0.HorzStride	
Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
Format:	HorzStride	
63:50	Dst.Operand	
Exists If:	([Dst.AddrMode]==Indirect)	
Format:	IndirectOperand	
63:50	Dst.Operand	
Exists If:	([Dst.AddrMode]==Direct)	
Format:	DirectOperand	
49:48	Dst.HorzStride	
Format:	HorzStride	
47	Reserved	
Format:	MBZ	
46	Src0.IsImm	
This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name

rnde - Round to Nearest or Even		
	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name Description
	0	Normal [Default] Normal. Per channel write enable used for final write enable generation.
	1	NoMask NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name Description

rnde - Round to Nearest or Even

	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	<p>PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>			Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>			Format:	PredCtrl							
Format:	PredCtrl											
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>											
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>											
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>			Format:	ChanOff							
Format:	ChanOff											
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>			Format:	ExecSize							
Format:	ExecSize											
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>			Format:	Header							
Format:	Header											

Round to Zero

rndz - Round to Zero			
Source:	Eulsa		
Length Bias:	4		
<p>The rndz instruction takes component-wise floating point round-to-zero operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-zero increments stored in the rounding increment bits. The round-to-zero increment must be added to the results in dst to create the final round-to-zero values to emulate the round-to-zero operation, commonly known as the truncate() function. The final results are the one of the two closest integral float values to the input values that is nearer to zero.</p>			
Format:	[(pred)] rndz[.cmod] (exec_size) dst src0		
Syntax			
[(pred)] rndz[.cmod] (exec_size) reg reg [(pred)] rndz[.cmod] (exec_size) reg imm32			
Pseudocode			
<pre>Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = floor(src0.chan[n]); if (abs(src0.chan[n]) < abs(dst.chan[n])) { dst.chan[n] = floor(src0.chan[n]) + 1; } } else { dst.chan[n] = floor(src0.chan[n]); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
F	F		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0]	
		Exists If:	([Src0.IsImm]==true)
	95:92	CondCtrl	
		Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))
		Format:	FlagModifier
95:64		Src0.ImmValue[63:32]	
		Exists If:	([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
87:84		Src0.VertStride	
		Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))

rndz - Round to Zero

	If:	([Src0.DataType]!=:df)	
	Format:	VertStride	
83:81	Src0.Width		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	Width	
80	Src0.AddrMode		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	

rndz - Round to Zero

45:44	Src0.Mod	Format:	SrcMod
43:40	Src0.DataType	Exists If:	([Src0.IsImm]==false)
		Format:	RegDataType
43:40	Src0.DataType	Exists If:	([Src0.IsImm]==true)
		Format:	ImmDataType
39:36	Dst.DataType	Format:	RegDataType
35	Dst.AddrMode	Format:	AddrMode
34	Saturate	Format:	Saturate
33	AccWrCtrl	Format:	AccWrCtrl
32	AtomicCtrl	Format:	AtomicCtrl
31	MaskCtrl	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
		Value	Name
		Description	
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
		Value	Name
		Description	
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.

rndz - Round to Zero

	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	<p>PredInv</p> <p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>			Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.										
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>			Format:	PredCtrl							
Format:	PredCtrl											
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>											
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>											
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>			Format:	ChanOff							
Format:	ChanOff											
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>			Format:	ExecSize							
Format:	ExecSize											
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>			Format:	Header							
Format:	Header											

Round Up

rndu - Round Up			
Source:	Eulsa		
Length Bias:	4		
<p>The rndu instruction takes component-wise floating point upward rounding (to the integral float number closer to positive infinity) of src0, commonly known as the ceiling() function. Each result follows the rules in the following tables based on the floating-point mode.</p>			
<p>Format:</p> <pre>[(pred)] rndu[.cmod] (exec_size) dst src0</pre>			
Syntax			
<pre>[(pred)] rndu[.cmod] (exec_size) reg reg [(pred)] rndu[.cmod] (exec_size) reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { if (src0.chan[n] - floor(src0.chan[n]) > 0.0f) { dst.chan[n] = floor(src0.chan[n]) + 1; } } else { dst.chan[n] = src0.chan[n]; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
F	F		
DWord	Bit	Description	
0..3	127:96	Src0.ImmValue[31:0] Exists If: ([Src0.IsImm]==true)	
	95:92	CondCtrl Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: FlagModifier	
	95:64	Src0.ImmValue[63:32] Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))	
	87:84	Src0.VertStride Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: VertStride	
	83:81	Src0.Width	

rndu - Round Up

	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	Width	
80	Src0.AddrMode		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct)	
	Format:	DirectOperand	
79:66	Src0.Operand		
	Exists If:	((([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Exists If:	([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))	
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Reserved		
	Format:	MBZ	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
45:44	Src0.Mod		
	Format:	SrcMod	
43:40	Src0.DataType		

rndu - Round Up

	Exists If:	([Src0.IsImm]==false)	
	Format:	RegDataType	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==true)	
	Format:	ImmDataType	
39:36	Dst.DataType		
	Format:	RegDataType	
35	Dst.AddrMode		
	Format:	AddrMode	
34	Saturate		
	Format:	Saturate	
33	AccWrCtrl		
	Format:	AccWrCtrl	
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl		
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".		
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:	MBZ	
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	This field, together with PredCtrl, enables and controls the generation of the predication mask		

rndu - Round Up

	<p>for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize							
Format:	ExecSize									
15:0	<p>Header</p> <table border="1"> <tr> <td>Format:</td> <td>Header</td> </tr> </table>	Format:	Header							
Format:	Header									

Sampler Cache Media Block Read MSD

MSD_SC_MB - Sampler Cache Media Block Read MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18	Reserved Format: _____ MBZ
	17:14	Message Type Default Value: _____ 05h Format: _____ Opcode Media Block Read Sampler Cache message
	13:11	Reserved Format: _____ MBZ
	10:8	Vertical Line Stride Override Format: _____ MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	Binding Table Index Format: _____ MDC_BTS Specifies the Binding Table Index for the message

Sampler Cache Oword Aligned Block Read MSD

MSD_SC_OWAB - Sampler Cache Oword Aligned Block Read MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18	Reserved Format: _____ MBZ
	17:14	Message Type Default Value: _____ 04h Format: _____ Opcode Oword Aligned Block Read Sampler Cache message
	13:11	Reserved Format: _____ MBZ
	10:8	Data Elements Format: _____ MDC_DB_OW Specifies the number of contiguous Owords to be read
	7:0	Binding Table Index Format: _____ MDC_BTS Specifies the Binding Table Index for the message

Scratch Block Read MSD

MSDOR_SB - Scratch Block Read MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18	Scratch Block Message Default Value: _____ 1h Format: _____ Opcode Scratch Block Message
	17	Operation Type Default Value: _____ 0h Format: _____ Opcode Scratch Block Read message
	16	Reserved Format: _____ MBZ
	15	Invalidate After Read Format: _____ MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	14	Reserved Format: _____ MBZ

MSDOR_SB - Scratch Block Read MSD			
13:12	Data Elements <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: center;">MDC_DB_HW</td> </tr> </table> <p>Specifies the number of registers to be read or written</p>	Format:	MDC_DB_HW
	Format:	MDC_DB_HW	
11:0	Address Offset <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>GeneralStateOffset[16:5]</td> </tr> </table> <p>DWORD (32 byte) based address offset to the BufferAddress in the Message Header.</p>	Format:	GeneralStateOffset[16:5]
	Format:	GeneralStateOffset[16:5]	

Scratch Block Write MSD

MSD0W_SB - Scratch Block Write MSD		
Source:		EuSubFunctionDataPort0
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: _____ MDC_MHR Indicates that the message requires a header.
	18	Scratch Block Message Default Value: _____ 1h Format: _____ Opcode Scratch Block Message
	17	Operation Type Default Value: _____ 1h Format: _____ Opcode Scratch Block Write message
	16:14	Reserved Format: _____ MBZ
	13:12	Data Elements Format: _____ MDC_DB_HW Specifies the number of registers to be read or written
	11:0	Address Offset Format: _____ GeneralStateOffset[17:6]

MSDOW_SB - Scratch Block Write MSD		
---	--	--

		WORD (16 byte) based address offset to the BufferAddress in the Message Header.
--	--	---

Select

sel - Select			
Source:	Eulsa		
Length Bias:	4		
<p>The sel instruction selectively moves the components in src0 or src1 into the channels of dst based on the predication. On a channel by channel basis, if the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst.</p> <p>As the predication is used to select the two sources, it is not included in the evaluation of WrEn. The predicate clause is mandatory if cmod is omitted/0000b. If both predication and the conditional modifier are omitted, the results are undefined.</p> <p>If the conditional modifier is specified (not 0000b, a compare is performed and the resulting condition flag is used for the sel instruction. Conditional modifiers .ge and .l follow the cmpn rules, and all other conditional modifiers follow the cmp rules. Predication is not allowed in this mode.</p> <p>A sel instruction with cmod .l is used to emulate a MIN instruction.</p> <p>A sel instruction with cmod .ge is used to emulate a MAX instruction.</p> <p>For a sel instruction with a .l or .ge conditional modifier, if one source is NaN and the other not NaN, the non-NaN source is the result. If both sources are NaNs, the result is NaN. For all other conditional modifiers, if either source is NaN then src1 is selected.</p> <p>A sel instruction without a conditional modifier always copies a denorm source value to a denorm destination value (in the manner of a raw move). This applies even if the source modifiers are set on the sel instruction sources.</p> <p>The sel instruction uses any conditional modifier internally and does not update the flag register if a conditional modifier is used.</p> <p>A sel instruction with cmod or source modifier will flush denorm to zero, depending on the denorm mode bit; a sel instruction without cmod and source modifier will retain denorm.</p>			
Format:	<pre>(pred) sel[.cmod] (exec_size) dst src0 src1</pre>		
Syntax			
<pre>(pred) sel[.cmod] (exec_size) reg reg reg (pred) sel[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate(WrEn, NoPMask); if (cmod == "0000") { // no CMod Evaluate(PMask); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { if (PMask.channel[n]) { dst.chan[n] = src0.chan[n]; } else { dst.chan[n] = src1.chan[n]; } } else { // with CMod Evaluate(CMod); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { if (CMod.chan[n]) { dst.chan[n] = src0.chan[n]; } else { dst.chan[n] = src1.chan[n]; } } } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y

sel - Select

Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
F	F		
HF	HF		

DWord	Bit	Description	
0.3	127:126	Reserved	
		Exists If:	((Src1.IsImm)==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
		Exists If:	((Src1.IsImm)==true)
	125:122	Reserved	
		Exists If:	((Src1.IsImm)==false)
		Format:	MBZ
	121:120	Src1.Mod	
		Exists If:	((Src1.IsImm)==false)
		Format:	SrcMod
	119:116	Src1.VertStride	
		Exists If:	((Src1.IsImm)==false)
		Format:	VertStride
115:113	Src1.Width		
	Exists If:	((Src1.IsImm)==false)	
	Format:	Width	
112	Src1.AddrMode		
	Exists If:	((Src1.IsImm)==false)	
	Format:	AddrMode	
111:98	Src1.Operand		
	Exists If:	((Src1.IsImm)==false) AND ((Src1.AddrMode)==Direct)	
	Format:	DirectOperand	
111:98	Src1.Operand		
	Exists If:	((Src1.IsImm)==false) AND ((Src1.AddrMode)==Indirect)	
	Format:	IndirectOperand	
97:96	Src1.HorzStride		
	Exists If:	((Src1.IsImm)==false)	
	Format:	HorzStride	
95:92	CondCtrl		
	Format:	FlagModifier	

sel - Select

	91:88	Src1.DataType	
		Exists If:	([Src1.IsImm]==false)
		Format:	RegDataType
	91:88	Src1.DataType	
		Exists If:	([Src1.IsImm]==true)
		Format:	ImmDataType
	87:84	Src0.VertStride	
		Format:	VertStride
	83:81	Src0.Width	
		Format:	Width
	80	Src0.AddrMode	
		Format:	AddrMode
	79:66	Src0.Operand	
		Exists If:	([Src0.AddrMode]==Direct)
		Format:	DirectOperand
	79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)	
	Format:	IndirectOperand	
65:64	Src0.HorzStride		
	Format:	HorzStride	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Direct)	
	Format:	DirectOperand	
63:50	Dst.Operand		
	Exists If:	([Dst.AddrMode]==Indirect)	
	Format:	IndirectOperand	
49:48	Dst.HorzStride		
	Format:	HorzStride	
47	Src1.IsImm		
	This field indicate that Source 1 operand is carrying an immediate value.		
	Value	Name	
	0	false [Default]	
	1	true	
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	

sel - Select		
	0	false [Default]
	1	true
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	((Src0.IsImm) == true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType
35	Dst.AddrMode	
	Format:	AddrMode
34	Saturate	
	Format:	Saturate
33	AccWrCtrl	
	Format:	AccWrCtrl
32	AtomicCtrl	
	Format:	AtomicCtrl
31	MaskCtrl	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name Description
	0	Normal [Default] Normal. Per channel write enable used for final write enable generation.
	1	NoMask NoMask. Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved	
29	CmptCtrl	
	Format:	MBZ
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name Description

sel - Select

	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff Format: ChanOff This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.		
18:16	ExecSize Format: ExecSize This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.		
15:0	Header Format: Header		

Send Message

send - Send Message	
Source:	Eulsa
Length Bias:	4
Description	
<p>The send instruction performs data communication between a thread and external function units, including shared functions (Data Port Read, Data Port Write, URB, and Message Gateway) and some fixed functions (e.g. Thread Spawner, who also have an unique Shared Function ID). The send instruction adds an entry to the EU's message request queue. The request message is stored in a split pair of contiguous GRF registers. Typically the header and addresses in one block and the data in another, but this is not strictly necessary and null may be passed as either parameter. The response message, if present, will be returned to a block of contiguous GRF registers. The return GRF writes may be in any order depending on the external function units. <src0> and <src1> are the lead GRF registers for the first and second block of the request respectively. <dest> is the lead GRF register for response. The message descriptor field <desc> contains the Message Length (the number of consecutive GRF registers corresponding to src0) and the Response Length (the number of consecutive GRF registers). It also contains the header present bit, and the function control signals. The extend message descriptor field <ex_desc> contains the target function ID, the Extended Message Length (the number of consecutive GRF registers corresponding to src1) and the extended function control signals. WrEn is forwarded to the target function in the message sideband. The send instruction is the only way to terminate a thread. When the EOT (End of Thread) bit of <ex_desc> is set, it indicates the end of thread to the EU, the Thread Dispatcher and, in most cases, the parent fixed function.</p>	
<p>The send instruction performs data communication between a thread and external function units, including shared functions (Sampler, Data Port Read, Data Port Write, URB, and Message Gateway) and some fixed functions (e.g. Thread Spawner, who also have an unique Shared Function ID). The send instruction adds an entry to the EU's message request queue. The request message is stored in a split pair of contiguous GRF registers. Typically the header and addresses in one block and the data in another, but this is not strictly necessary and null may be passed as either parameter. The response message, if present, will be returned to a block of contiguous GRF registers. The return GRF writes may be in any order depending on the external function units. <src0> and <src1> are the lead GRF registers for the first and second block of the request respectively. <dest> is the lead GRF register for response. The message descriptor field <desc> contains the Message Length (the number of consecutive GRF registers corresponding to src0) and the Response Length (the number of consecutive GRF registers). It also contains the header present bit, and the function control signals. The extend message descriptor field <ex_desc> contains the target function ID, the Extended Message Length (the number of consecutive GRF registers corresponding to src1) and the extended function control signals. WrEn is forwarded to the target function in the message sideband. The send instruction is the only way to terminate a thread. When the EOT (End of Thread) bit of <ex_desc> is set, it indicates the end of thread to the EU, the Thread Dispatcher and, in most cases, the parent fixed function.</p>	
<p>Message descriptor field <desc> can be a 32-bit immediate, imm32, or a 32-bit scalar register, <reg32a>. The 32-bit scalar register <reg32a> must be the leading dword of the address register. It should be in the form of a0.0.</p>	
<p><src0> is a GRF register. It serves as the leading GRF register of the request. <src1> is a GRF register or a null register. It serves as the leading GRF register for the second block of the request when it is not a null register. It is required that the second block of GRFs does not overlap with the first</p>	

send - Send Message

block. If it is a null register the Extended Message Length must be 0. The sum of Message Length and Extended Message Length must not be greater than 15.

<dest> serves for two purposes: to provide the leading GRF register location for the response message if present, and to provide parameters to form the channel-enable sideband signals.

<dest> signals whether there is a response to the message request. It can be either a null register or a GRF register.

If <dest> is null, there is no response to the request. Meanwhile, the Response Length field in <desc> must be 0. Certain types of message requests, such as memory write (store) through the Data Port, do not want response data from the function unit. If so, the posted destination operand can be null.

If <dest> is a GRF register, the register number is forwarded to the shared function. In this case, the target function unit must send one or more response message phases back to the requesting thread. The number of response message phases must match the Response Length field in <desc>, which of course cannot be zero. For some cases, it could be an empty return message. An empty return message is defined as a single phase message with all channel enables turned off.

The 16-bit channel enables of the message sideband are formed based on the WrEn. Interpretation of the channel-enable sideband signals is subject to the target external function. In general for a 'send' instruction with return messages, they are used as the destination dword write mask for the GRF registers starting at <dest>. For a message that has multiple return phases, the same set of channel enable signals applies to all the return phases.

Send a message stored in GRF locations starting at <src0> followed by <src1> to a shared function identified by <ex_desc> along with control from <desc> and <ex_desc> with a GRF writeback location at <dest>.

Format:

```
[ (pred) ] send.<sfid> (exec_size) <dest> <src0> <src1> <ex_desc> <desc> {[EOT]}
```

Programming Notes

Send instruction execution (reading GRFs and sending out) is guaranteed to be in-order for a SharedFunction specified by SFID except for SLM. SLM SharedFunction is decoded as follows.

```
SLM = ((SFID==DC0) && (desc[18] == 0) && (desc[7:0]==0xFE)) ||
((SFID==DC1) && (desc[7:0]==0xFE)) ||
((SFID==DC2) && (desc[19] == 0)&&(desc[7]==0x1))
```

Restriction

Software must obey the following rules in signaling the end-of-thread using the send instruction: The posted destination operand must be null. No acknowledgement is allowed for the send instruction that signifies the end of thread. This is to avoid deadlock as the EU is expecting to free up the terminated thread's resource. A thread must terminate with a send instruction with message to a shared function on the output message bus; therefore, it cannot terminate with a send instruction with message to the NULL function. For example, a thread may terminate with a URB write message or a render cache write message. A root thread originated from the media (generic) pipeline must terminate with a send instruction with message to the Thread Spawner unit. A child thread should also terminate with a send to TS. Please refer to the Media Chapter for more detailed description.

Software must obey the following rules in signaling the end-of-thread using the send instruction: The posted destination operand must be null. No acknowledgement is allowed for the send instruction that signifies the end of thread. This is to avoid deadlock as the EU is expecting to free up the terminated thread's resource. A thread must terminate with a send instruction with message to a shared function on the output message bus;

send - Send Message

therefore, it cannot terminate with a send instruction with message to the sampler unit or the NULL function. For example, a thread may terminate with a URB write message or a render cache write message. A root thread originated from the media (generic) pipeline must terminate with a send instruction with message to the Thread Spawner unit. A child thread should also terminate with a send to TS. Please refer to the Media Chapter for more detailed description.

A send instruction can not update accumulator registers.

EOT must NOT be set for the send instruction when using indirect register addressing mode.

An EOT send must use register space r112-r127 for sources. This is to enable loading of a new thread into the same slot while the message with EOT for current thread is pending dispatch.

Syntax

```
[(pred)] send.<sfid> (exec_size) reg greg reg (imm32|reg32a) (imm32|reg32a)
{ [EOT] }
```

Pseudocode

```
Evaluate (WrEn);           <MsgChEnable> = WrEn;           <SourceReg0> = <src0>.RegNum;
<SourceReg1> = <src1>.RegNum;   MessageEnqueue (<MsgChEnable>, <ResponseReg>,
<SourceReg0>, <SourceReg1>, <ex_desc>, <dest>);
```

Predication	Conditional Modifier	Saturation	Source Modifier	Subfunctions []
Y	N	N	N	Unspecified

DWord	Bit	Description
0..3	127:124	ExDesc[31:28]
		Exists If: ([ExDesc.IsReg]==false)
	Format: ExMsgDesc[31:28]	
	127:124	Reserved
		Exists If: ([ExDesc.IsReg]==true)
	Format: MBZ	
123:122	Desc[31:30]	
	Exists If: ([Desc.IsReg]==false)	
Format: MsgDesc[31:30]		
123:113	Reserved	
	Exists If: ([Desc.IsReg]==true)	
Format: MBZ		
121:113	Desc[19:11]	
	Exists If: ([Desc.IsReg]==false)	
Format: MsgDesc[19:11]		
112	Reserved	
	Format: MBZ	

send - Send Message

111:104	Src1.RegNum	Format:	DirectOperand[13:6]
103:99	Src1.Length	Exists If:	([ExDesc.IsReg]==false)
	Format:	ExMsgDesc[10:6]	
103:99	Reserved	Exists If:	([ExDesc.IsReg]==true)
	Format:	MBZ	
98	Src1.RegFile	Format:	DirectOperand[0]
97:96	ExDesc[27:26]	Exists If:	([ExDesc.IsReg]==false)
	Format:	ExMsgDesc[27:26]	
97:96	Reserved	Exists If:	([ExDesc.IsReg]==true)
	Format:	MBZ	
95:92	SFID	Format:	SFID
91:81	Desc[10:0]	Exists If:	([Desc.IsReg]==false)
	Format:	MsgDesc[10:0]	
91:81	Reserved	Exists If:	([Desc.IsReg]==true)
	Format:	MBZ	
80	Reserved	Format:	MBZ
79:72	Src0.RegNum	Format:	DirectOperand[13:6]
71	MsgDesc[29]	Exists If:	([Desc.IsReg]==false)
	Format:	MsgDesc[29]	
71:67	Reserved	Exists If:	([Desc.IsReg]==true)
	Format:	MBZ	
70:67	Src0.Length	Exists If:	([Desc.IsReg]==false)

send - Send Message

		Format:	MsgDesc[28:25]
66	Src0.RegFile		
		Format:	DirectOperand[0]
65:64	ExDesc[25:24]		
	Exists If:	([ExDesc.IsReg]==false)	
	Format:	ExMsgDesc[25:24]	
65:64	Reserved		
	Exists If:	([ExDesc.IsReg]==true)	
	Format:	MBZ	
63:56	Dst.RegNum		
	Format:	DirectOperand[13:6]	
55:51	Dst.Length		
	Exists If:	([Desc.IsReg]==false)	
	Format:	MsgDesc[24:20]	
55:51	Reserved		
	Exists If:	([Desc.IsReg]==true)	
	Format:	MBZ	
50	Dst.RegFile		
	Format:	DirectOperand[0]	
49	ExDesc.IsReg	This field indicates that the extended message descriptor is provided by the address register, selected by the AddrSubRegNum[3:1].	
		Value	Name
		0	false
		1	true
48	Desc.IsReg	This field indicates that the message descriptor is provided by the address subregister a0.0.	
		Value	Name
		0	false
		1	true
47:43	Reserved		
	Exists If:	([ExDesc.IsReg]==true)	
	Format:	MBZ	
47:35	ExDesc[23:11]		
	Exists If:	([ExDesc.IsReg]==false)	
	Format:	ExMsgDesc[23:11]	

send - Send Message

	42:40	AddrSubRegNum[3:1]	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == true)</td> </tr> <tr> <td>Format:</td> <td>AddrSubRegNum[3:1]</td> </tr> </table>	Exists If:	([ExDesc.IsReg] == true)	Format:	AddrSubRegNum[3:1]					
Exists If:	([ExDesc.IsReg] == true)											
Format:	AddrSubRegNum[3:1]											
	38:36	Reserved	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == true)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([ExDesc.IsReg] == true)	Format:	MBZ					
Exists If:	([ExDesc.IsReg] == true)											
Format:	MBZ											
	35	Reserved	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == true)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([ExDesc.IsReg] == true)	Format:	MBZ					
Exists If:	([ExDesc.IsReg] == true)											
Format:	MBZ											
	34	EOT	<p>This field controls the termination of the thread. For a send instruction, if this field is set, EU will terminate the thread and also set the EOT bit in the message sideband. This field only applies to the send instruction. It is not present for other instructions.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Thread is not terminated</td> </tr> <tr> <td style="text-align: center;">1</td> <td>EOT</td> </tr> </tbody> </table>	Value	Name	0	Thread is not terminated	1	EOT			
Value	Name											
0	Thread is not terminated											
1	EOT											
	33	FusionCtrl	<p>This field provides explicit control for EU fusion lock-step execution. When this bit is set to 1b, the instruction is executed serially starting from the first EU to the last EU in the fused set.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal lockstep execution</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Serialized execution</td> </tr> </tbody> </table>	Value	Name	0	Normal lockstep execution	1	Serialized execution			
Value	Name											
0	Normal lockstep execution											
1	Serialized execution											
	32	AtomicCtrl	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>AtomicCtrl</td> </tr> </table>	Format:	AtomicCtrl							
Format:	AtomicCtrl											
	31	MaskCtrl	<p>Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0".</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description										
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.										
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.										
	30	Reserved										
	29	CmptCtrl	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table> <p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations</p>	Format:	MBZ							
Format:	MBZ											

send - Send Message

	supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.									
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.								
28	<p>PredInv</p> <p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0]</p> <p>This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									
18:16	<p>ExecSize</p> <table border="1"> <tr> <td>Format:</td> <td>ExecSize</td> </tr> </table>	Format:	ExecSize							
Format:	ExecSize									

send - Send Message		
		This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	Header
		Format: Header

Send Message Conditional

sendc - Send Message Conditional				
Source:	Eulsa			
Length Bias:	4			
<p>The sendc instruction has the same behavior as the sends instruction except the following. sendc first checks the dependent threads inside the Thread Dependency Register. There are up to 8 dependent threads in the TDR register. The sendc instruction executes only when all the dependent threads in the TDR register are retired.</p> <p>Wait for dependencies in the TDR Register to clear, then send a message stored in GRF locations starting at <src0> followed by <src1> to a shared function identified by <ex_desc> along with control from <desc> and <ex_desc> with a GRF writeback location at <dest>.</p> <p>Format:</p> <pre>[(pred)] sendc.<sfid> (exec_size) <dest> <src0> <src1> <ex_desc> <desc> {[EOT]}</pre>				
Restriction				
The sendc instruction has the same restrictions as the send instruction.				
Syntax				
<pre>[(pred)] sendc.<sfid> (exec_size) reg greg reg (imm32 reg32a) (imm32 reg32a) {[EOT]}</pre>				
Pseudocode				
<pre>if (TDR[7] ... TDR[2] TDR[1] TDR[0]) { wait; } Evaluate(WrEn); <MsgChEnable> = WrEn; MessageEnqueue(<MsgChEnable>, <ResponseReg>, <src0>.RegNum, <src1>.RegNum; ,<ex_desc>, <dest>.RegNum);</pre>				
Predication	Conditional Modifier	Saturation	Source Modifier	Subfunctions []
Y	N	N	N	Unspecified
DWord	Bit	Description		
0..3	127:124	ExDesc[31:28]		
		Exists If:	([ExDesc.IsReg]==false)	
	Format:		ExMsgDesc[31:28]	
	127:124	Reserved		
Exists If:		([ExDesc.IsReg]==true)		
Format:		MBZ		
123:122	Desc[31:30]			
	Exists If:	([Desc.IsReg]==false)		
Format:		MsgDesc[31:30]		
123:113	Reserved			

sendc - Send Message Conditional

	Exists If:	([Desc.IsReg]==true)	
	Format:	MBZ	
121:113	Desc[19:11]		
	Exists If:	([Desc.IsReg]==false)	
	Format:	MsgDesc[19:11]	
112	Reserved		
	Format:	MBZ	
111:104	Src1.RegNum		
	Format:	DirectOperand[13:6]	
103:99	Src1.Length		
	Exists If:	([ExDesc.IsReg]==false)	
	Format:	ExMsgDesc[10:6]	
103:99	Reserved		
	Exists If:	([ExDesc.IsReg]==true)	
	Format:	MBZ	
98	Src1.RegFile		
	Format:	DirectOperand[0]	
97:96	ExDesc[27:26]		
	Exists If:	([ExDesc.IsReg]==false)	
	Format:	ExMsgDesc[27:26]	
97:96	Reserved		
	Exists If:	([ExDesc.IsReg]==true)	
	Format:	MBZ	
95:92	SFID		
	Format:	SFID	
91:81	Desc[10:0]		
	Exists If:	([Desc.IsReg]==false)	
	Format:	MsgDesc[10:0]	
91:81	Reserved		
	Exists If:	([Desc.IsReg]==true)	
	Format:	MBZ	
80	Reserved		
	Format:	MBZ	
79:72	Src0.RegNum		
	Format:	DirectOperand[13:6]	
71	MsgDesc[29]		

sendc - Send Message Conditional

		Exists If:	([Desc.IsReg]==false)
		Format:	MsgDesc[29]
71:67	Reserved		
		Exists If:	((Desc.IsReg)==true)
		Format:	MBZ
70:67	Src0.Length		
		Exists If:	((Desc.IsReg)==false)
		Format:	MsgDesc[28:25]
66	Src0.RegFile		
		Format:	DirectOperand[0]
65:64	ExDesc[25:24]		
		Exists If:	((ExDesc.IsReg)==false)
		Format:	ExMsgDesc[25:24]
65:64	Reserved		
		Exists If:	((ExDesc.IsReg)==true)
		Format:	MBZ
63:56	Dst.RegNum		
		Format:	DirectOperand[13:6]
55:51	Dst.Length		
		Exists If:	((Desc.IsReg)==false)
		Format:	MsgDesc[24:20]
55:51	Reserved		
		Exists If:	((Desc.IsReg)==true)
		Format:	MBZ
50	Dst.RegFile		
		Format:	DirectOperand[0]
49	ExDesc.IsReg		
	This field indicates that the extended message descriptor is provided by the address register, selected by the AddrSubRegNum[3:1].		
	Value	Name	
	0	false	
	1	true	
48	Desc.IsReg		
	This field indicates that the message descriptor is provided by the address subregister a0.0.		
	Value	Name	
	0	false	

sendc - Send Message Conditional

	1	true									
47:43	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == true)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>		Exists If:	([ExDesc.IsReg] == true)	Format:	MBZ					
Exists If:	([ExDesc.IsReg] == true)										
Format:	MBZ										
47:35	ExDesc[23:11] <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == false)</td> </tr> <tr> <td>Format:</td> <td>ExMsgDesc[23:11]</td> </tr> </table>		Exists If:	([ExDesc.IsReg] == false)	Format:	ExMsgDesc[23:11]					
Exists If:	([ExDesc.IsReg] == false)										
Format:	ExMsgDesc[23:11]										
42:40	AddrSubRegNum[3:1] <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == true)</td> </tr> <tr> <td>Format:</td> <td>AddrSubRegNum[3:1]</td> </tr> </table>		Exists If:	([ExDesc.IsReg] == true)	Format:	AddrSubRegNum[3:1]					
Exists If:	([ExDesc.IsReg] == true)										
Format:	AddrSubRegNum[3:1]										
38:36	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == true)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>		Exists If:	([ExDesc.IsReg] == true)	Format:	MBZ					
Exists If:	([ExDesc.IsReg] == true)										
Format:	MBZ										
35	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Exists If:</td> <td>([ExDesc.IsReg] == true)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>		Exists If:	([ExDesc.IsReg] == true)	Format:	MBZ					
Exists If:	([ExDesc.IsReg] == true)										
Format:	MBZ										
34	EOT <p>This field controls the termination of the thread. For a send instruction, if this field is set, EU will terminate the thread and also set the EOT bit in the message sideband. This field only applies to the send instruction. It is not present for other instructions.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Thread is not terminated</td> </tr> <tr> <td style="text-align: center;">1</td> <td>EOT</td> </tr> </tbody> </table>		Value	Name	0	Thread is not terminated	1	EOT			
Value	Name										
0	Thread is not terminated										
1	EOT										
33	FusionCtrl <p>This field provides explicit control for EU fusion lock-step execution. When this bit is set to 1b, the instruction is executed serially starting from the first EU to the last EU in the fused set.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal lockstep execution</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Serialized execution</td> </tr> </tbody> </table>		Value	Name	0	Normal lockstep execution	1	Serialized execution			
Value	Name										
0	Normal lockstep execution										
1	Serialized execution										
32	AtomicCtrl <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 45%;">Format:</td> <td>AtomicCtrl</td> </tr> </table>		Format:	AtomicCtrl							
Format:	AtomicCtrl										
31	MaskCtrl <p>Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0".</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask.Skips the check for PciP[n] == ExIP before enabling a channel,</td> </tr> </tbody> </table>		Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel,
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask.Skips the check for PciP[n] == ExIP before enabling a channel,									

sendc - Send Message Conditional

			as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:		MBZ
	<p>Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p>		
	Value	Name	Description
	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.
	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	PredInv		
	<p>This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p>		
	Value	Name	Description
	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.
	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	PredCtrl		
	Format:		PredCtrl
	<p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>		
23	FlagRegNum[0]		
	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum		
	<p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		

sendc - Send Message Conditional			
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

SFC_AVS_CHROMA_Coeff_Table

SFC_AVS_CHROMA_Coeff_Table							
Source:	BSpec						
Length Bias:	2						
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.							
DWord	Bit	Description					
0	31:29	Command Type					
		Default Value:	3h PARALLEL_VIDEO_PIPE				
		Format:	OpCode				
	28:27	Pipeline					
		Default Value:	2h Media				
	26:23	Media Command Opcode					
		Format:	OpCode				
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Ah</td> <td>Media Misc [Default]</td> <td>[] Media MFX/VEBOX+SFC Mode</td> </tr> </tbody> </table>	Value	Name	Description	Ah	Media Misc [Default]
	Value	Name	Description				
	Ah	Media Misc [Default]	[] Media MFX/VEBOX+SFC Mode				
22:21	SubOpcodeA						
	Default Value:	0h Common					
20:16	SubOpcodeB						
	Default Value:	6h SFC_AVS CHROMA Coeff_Table					
15:12	Reserved						
	Format:	MBZ					
11:0	DWord Length						
	Default Value:	3Fh Excludes DWord (0,1)					
	Format:	=n					
Total Length - 2							
1.64	2047:0	AVS CHROMA Coefficient Table Body					
		Format:	SFC_AVS_CHROMA_COEFF_TABLE_BODY				

SFC_AVS_LUMA_Coeff_Table

SFC_AVS_LUMA_Coeff_Table				
Source:	BSpec			
Length Bias:	2			
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	Pipeline		
		Default Value:	2h Media	
		Format:	OpCode	
	26:23	Media Command Opcode		
		Format:	OpCode	
		Value	Name	Description
		Ah	Media Misc [Default]	[] Media MFX/VEBOX+SFC Mode
22:21	SubOpcodeA			
	Default Value:	0h Common		
	Format:	OpCode		
20:16	SubOpcodeB			
	Default Value:	5h SFC_AVS LUMA Coeff_Table		
	Format:	OpCode		
15:12	Reserved			
	Format:	MBZ		
11:0	DWord Length			
	Default Value:	7Fh Excludes DWord (0,1)		
	Format:	=n		
	Total Length - 2			
1..128	4095:0	AVS LUMA Coefficient Table Body		
		Format:	SFC_AVS_LUMA_COEFF_TABLE_BODY	

SFC_AVS_STATE

SFC_AVS_STATE							
Source:	BSpec						
Length Bias:	2						
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.							
DWord	Bit	Description					
0	31:29	Command Type					
		Default Value:	3h PARALLEL_VIDEO_PIPE				
		Format:	OpCode				
	28:27	Pipeline					
		Default Value:	2h Media				
	26:23	Media Command Opcode					
		Format:	OpCode				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Ah</td> <td>Media Misc [Default]</td> <td>Media MFX/VEBOX+SFC Modegen</td> </tr> </tbody> </table>	Value	Name	Description	Ah	Media Misc [Default]
	Value	Name	Description				
	Ah	Media Misc [Default]	Media MFX/VEBOX+SFC Modegen				
22:21	SubOpcodeA						
	Default Value:	0h Common					
20:16	SubOpcodeB						
	Default Value:	2h SFC_AVS_STATE					
15:12	Reserved						
	Format:	MBZ					
11:0	DWord Length						
	Default Value:	2h Excludes DWord (0,1)					
	Format:	=n					
	Total Length - 2						
1..3	95:0	AVS State Body					
		Format:	SFC_AVS_STATE_BODY				

SFC_FRAME_START

SFC_FRAME_START							
Source:	BSpec						
Length Bias:	2						
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.							
DWord	Bit	Description					
0	31:29	Command Type					
		Default Value:	3h PARALLEL_VIDEO_PIPE				
		Format:	OpCode				
	28:27	Pipeline					
		Default Value:	2h Media				
	26:23	Media Command Opcode					
		Format:	OpCode				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 45%;">Description</th> </tr> </thead> <tbody> <tr> <td>Ah</td> <td>Media Misc [Default]</td> <td>[] Media MFX/VEBOX+SFC Mode</td> </tr> </tbody> </table>	Value	Name	Description	Ah	Media Misc [Default]
	Value	Name	Description				
	Ah	Media Misc [Default]	[] Media MFX/VEBOX+SFC Mode				
22:21	SubOpcodeA						
	Default Value:	0h Common					
20:16	SubOpcodeB						
	Default Value:	4h SFC_FRAME_START					
15:12	Reserved						
	Format:	MBZ					
11:0	DWord Length						
	Default Value:	0h Excludes DWord (0,1)					
	Format:	=n					
Total Length - 2							
1	31:0	Frame Start Body					
		Format:	SFC_FRAME_START_BODY				

SFC_IEF_STATE

SFC_IEF_STATE									
Source:	BSpec								
Length Bias:	2								
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.									
DWord	Bit	Description							
0	31:29	Command Type							
		Default Value:	3h PARALLEL_VIDEO_PIPE						
		Format:	OpCode						
	28:27	Pipeline							
		Default Value:	2h Media						
	26:23	Media Command Opcode							
		Format:	OpCode						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Ah</td> <td>Media Misc [Default]</td> <td><input type="checkbox"/> Media MFX/VEBOX+SFC Mode</td> </tr> </tbody> </table>			Value	Name	Description	Ah	Media Misc [Default]	<input type="checkbox"/> Media MFX/VEBOX+SFC Mode
	Value	Name	Description						
	Ah	Media Misc [Default]	<input type="checkbox"/> Media MFX/VEBOX+SFC Mode						
22:21	SubOpcodeA								
	Default Value:	0h Common							
20:16	SubOpcodeB								
	Default Value:	3h SFC_IEF_STATE							
15:12	Reserved								
	Format:	MBZ							
11:0	DWord Length								
	Default Value:	16h Excludes DWord (0,1)							
	Format:	=n							
Total Length - 2									
1..23	735:0	SFC IEF State Body							
		Format:	SFC_IEF_STATE_BODY						

SFC_LOCK

SFC_LOCK				
Source:	BSpec			
Length Bias:	2			
Description				
<p>This command is used for VD/VE box to communicate with SFC before the start of any SFC workload. VD/VE uses this command to make sure that it has the ownership of SFC pipe before running workload with SFC since SFC is shared between VD/VE on a frame level.</p> <p>For VD(MFX)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC.</p> <p>For VD(HCP)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value:	3h PARALLEL_VIDEO_PIPE	
		Format:	OpCode	
	28:27	Pipeline		
		Default Value:	2h Media	
		Format:	OpCode	
	26:23	Media Command Opcode		
		Format:	OpCode	
		Value	Name	Description
		Ah	Media Misc [Default]	<p>Media MFX/VEBOX+SFC Mode</p> <p>For VD(MFX)+SFC mode, only decoder mode is allowed. Encoder mode cannot use SFC</p>
22:21	SubOpcodeA			
	Default Value:	0h Common		
	Format:	OpCode		
20:16	SubOpcodeB			
	Default Value:	0h SFC Lock		
	Format:	OpCode		
15:12	Reserved			
	Format:	MBZ		
11:0	DWord Length			
	Default Value:	0h Excludes DWord (0,1)		
	Format:	=n		
	Total Length - 2			

SFC_LOCK		
1	31:0	SFC Lock Body
		Format: SFC_LOCK_BODY

SFC_STATE

SFC_STATE		
Source:	BSpec	
Length Bias:	2	
This command is sent from VDBOX/HCP/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h PARALLEL_VIDEO_PIPE
		Format: OpCode
	28:27	Pipeline
		Default Value: 2h Media Format: OpCode
	26:23	Media Command Opcode
		Default Value: Ah Media MFX/VEBOX+SFC Mode Format: OpCode
	22:21	SubOpcodeA
Default Value: 0h Common Format: OpCode		
20:16	SubOpcodeB	
	Default Value: 1h SFC_State Format: OpCode	
15:12	Reserved	
11:0	DWord Length	
	Default Value: 2Bh Excludes DWord (0,1)	
	Format: =n	
	Total Length - 2	
1..49	1567:0	SFC State Body
		Format: SFC_STATE_BODY

Shift Left

shl - Shift Left			
Source:	Eulsa		
Length Bias:	4		
<p>Perform component-wise logical left shift of the bits in src0 by the shift count indicated in src1, storing the results in dst, inserting zero bits in the number of LSBs indicated by the shift count. Hardware detects overflow properly and uses it to perform any saturation operation on the result, as long as the shifted result is within 33 bits. Otherwise, the result is undefined. Note: For word and DWord operands, the accumulators have 33 bits.</p> <p>In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.</p>			
Format:	[(pred)] shl[.cmod] (exec_size) dst src0 src1		
Restriction			
Accumulator cannot be destination, implicit or explicit.			
Syntax			
[(pred)] shl[.cmod] (exec_size) reg reg reg [(pred)] shl[.cmod] (exec_size) reg reg imm32			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] & 0x1F dst.chan[n] = src0.chan[n] << shiftCnt; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
*B,*W,*D	*B,*W,*D		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
	125:122	Reserved	
		Exists If:	([Src1.IsImm]==false)

shl - Shift Left

	Format:	MBZ
121:120	Src1.Mod	
	Exists If:	((Src1.IsImm) == false)
	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	((Src1.IsImm) == false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	((Src1.IsImm) == false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode

shl - Shift Left

79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	DirectOperand
79:66	Src0.Operand	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	IndirectOperand
65:64	Src0.HorzStride	
	Format:	HorzStride
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	DirectOperand
63:50	Dst.Operand	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	IndirectOperand
49:48	Dst.HorzStride	
	Format:	HorzStride
47	Src1.IsImm	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
1	true	
46	Src0.IsImm	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name
	0	false [Default]
1	true	
45:44	Src0.Mod	
	Format:	SrcMod
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==false)
	Format:	RegDataType
43:40	Src0.DataType	
	Exists If:	([Src0.IsImm]==true)
	Format:	ImmDataType
39:36	Dst.DataType	
	Format:	RegDataType

shl - Shift Left

35	Dst.AddrMode	Format: AddrMode										
34	Saturate	Format: Saturate										
33	AccWrCtrl	Format: AccWrCtrl										
32	AtomicCtrl	Format: AtomicCtrl										
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 20%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>		Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description										
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.										
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.										
30	Reserved											
29	CmptCtrl Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 20%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>		Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description										
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.										
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.										
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th style="width: 20%;">Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the
Value	Name	Description										
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.										
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the										

shl - Shift Left

			predication mask.
27:24	PredCtrl	Format:	PredCtrl
	This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0]	This field specifies bit[0] of the register number for a flag register operand.	
22	FlagSubRegNum	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.	
21:19	ChanOff	Format:	ChanOff
	This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.		
18:16	ExecSize	Format:	ExecSize
	This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.		
15:0	Header	Format:	Header

Shift Right

shr - Shift Right			
Source:	Eulsa		
Length Bias:	4		
<p>Perform component-wise logical right shift with zero insertion of the bits in src0 by the shift count indicated in src1, storing the results in dst. Insert zero bits in the number of MSBs indicated by the shift count. Note: For word and DWord operands, the accumulators have 33 bits. Note: For unsigned src0 types, shr and asr produce the same result.</p> <p>In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.</p>			
<p>Format:</p> <pre>[(pred)] shr[.cmod] (exec_size) dst src0 src1</pre>			
Syntax			
<pre>[(pred)] shr[.cmod] (exec_size) reg reg reg [(pred)] shr[.cmod] (exec_size) reg reg imm32</pre>			
Pseudocode			
<pre>Evaluate (WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] & 0x1F dst.chan[n] = src0.chan[n] » shiftCnt; } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	Y
Src Types	Dst Types		
UB, UW, UD	UB, UW, UD		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
	127:96	Src1.ImmValue[31:0]	
		Exists If:	([Src1.IsImm]==true)
	125:122	Reserved	
		Exists If:	([Src1.IsImm]==false)
	Format:	MBZ	
121:120	Src1.Mod		
	Exists If:	([Src1.IsImm]==false)	

shr - Shift Right

	Format:	SrcMod
119:116	Src1.VertStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	VertStride
115:113	Src1.Width	
	Exists If:	((Src1.IsImm) == false)
	Format:	Width
112	Src1.AddrMode	
	Exists If:	((Src1.IsImm) == false)
	Format:	AddrMode
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Direct)
	Format:	DirectOperand
111:98	Src1.Operand	
	Exists If:	((Src1.IsImm) == false) AND ((Src1.AddrMode) == Indirect)
	Format:	IndirectOperand
97:96	Src1.HorzStride	
	Exists If:	((Src1.IsImm) == false)
	Format:	HorzStride
95:92	CondCtrl	
	Format:	FlagModifier
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == false)
	Format:	RegDataType
91:88	Src1.DataType	
	Exists If:	((Src1.IsImm) == true)
	Format:	ImmDataType
87:84	Src0.VertStride	
	Format:	VertStride
83:81	Src0.Width	
	Format:	Width
80	Src0.AddrMode	
	Format:	AddrMode
79:66	Src0.Operand	
	Exists If:	((Src0.AddrMode) == Direct)
	Format:	DirectOperand

shr - Shift Right

79:66	Src0.Operand	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">Exists If:</td> <td>([Src0.AddrMode]==Indirect)</td> </tr> <tr> <td>Format:</td> <td>IndirectOperand</td> </tr> </table>	Exists If:	([Src0.AddrMode]==Indirect)	Format:	IndirectOperand		
Exists If:	([Src0.AddrMode]==Indirect)							
Format:	IndirectOperand							
65:64	Src0.HorzStride	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>HorzStride</td> </tr> </table>	Format:	HorzStride				
Format:	HorzStride							
63:50	Dst.Operand	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">Exists If:</td> <td>([Dst.AddrMode]==Direct)</td> </tr> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Exists If:	([Dst.AddrMode]==Direct)	Format:	DirectOperand		
Exists If:	([Dst.AddrMode]==Direct)							
Format:	DirectOperand							
63:50	Dst.Operand	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">Exists If:</td> <td>([Dst.AddrMode]==Indirect)</td> </tr> <tr> <td>Format:</td> <td>IndirectOperand</td> </tr> </table>	Exists If:	([Dst.AddrMode]==Indirect)	Format:	IndirectOperand		
Exists If:	([Dst.AddrMode]==Indirect)							
Format:	IndirectOperand							
49:48	Dst.HorzStride	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>HorzStride</td> </tr> </table>	Format:	HorzStride				
Format:	HorzStride							
47	Src1.IsImm	<p>This field indicate that Source 1 operand is carrying an immediate value.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>false [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </tbody> </table>	Value	Name	0	false [Default]	1	true
Value	Name							
0	false [Default]							
1	true							
46	Src0.IsImm	<p>This field indicate that Source 0 operand is carrying an immediate value.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>false [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </tbody> </table>	Value	Name	0	false [Default]	1	true
Value	Name							
0	false [Default]							
1	true							
45:44	Src0.Mod	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>SrcMod</td> </tr> </table>	Format:	SrcMod				
Format:	SrcMod							
43:40	Src0.DataType	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>RegDataType</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	RegDataType		
Exists If:	([Src0.IsImm]==false)							
Format:	RegDataType							
43:40	Src0.DataType	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">Exists If:</td> <td>([Src0.IsImm]==true)</td> </tr> <tr> <td>Format:</td> <td>ImmDataType</td> </tr> </table>	Exists If:	([Src0.IsImm]==true)	Format:	ImmDataType		
Exists If:	([Src0.IsImm]==true)							
Format:	ImmDataType							
39:36	Dst.DataType	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>RegDataType</td> </tr> </table>	Format:	RegDataType				
Format:	RegDataType							
35	Dst.AddrMode	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>AddrMode</td> </tr> </table>	Format:	AddrMode				
Format:	AddrMode							
34	Saturate							

shr - Shift Right

	Format:	Saturate									
33	AccWrCtrl										
	Format:	AccWrCtrl									
32	AtomicCtrl										
	Format:	AtomicCtrl									
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td>1</td> <td>NoMask</td> <td>NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>	Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.									
30	Reserved										
29	CmptCtrl										
	Format:	MBZ									
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.	
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.	
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	PredCtrl										
	Format:	PredCtrl									

shr - Shift Right

		This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.		
23	FlagRegNum[0]	This field specifies bit[0] of the register number for a flag register operand.		
22	FlagSubRegNum	This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.		
21:19	ChanOff	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff			
18:16	ExecSize	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize			
15:0	Header	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header			

Signal Event

MSD_SIGNAL_EVENT - Signal Event							
Source:	EuSubFunctionGateway						
Length Bias:	1						
Sends an event to all threads that are monitoring that Event ID.							
DWord	Bit	Description					
0	31:29	Reserved					
		Format: MBZ					
	28:25	Message Length					
		Format: U4					
		Specifies the number of GRF registers sent as the message payload.					
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>One [Default]</td> <td>See MDP_EVENT Event Data payload definition.</td> </tr> </tbody> </table>	Value	Name	Description	1	One [Default]
	Value	Name	Description				
	1	One [Default]	See MDP_EVENT Event Data payload definition.				
	24:20	Response Length					
		Default Value: 0 None					
Format: U5							
Specifies the number of GRF registers expected as the message response payload.							
19:3	Reserved						
	Format: MBZ						
2:0	Signal Event Subfunction						
	Default Value: 0x1						
	Format: OpCode						

SIMD8 Render Target Read MSD

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 0h Format: Opcode Full precision data message Programming Notes This field must be programmed to 0
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Reserved Format: MBZ
	17:14	Message Type Default Value: 0Dh Format: Opcode Render Target Read message
	13	Per-Sample PS Enable Format: Enable

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD

		<p>If set, PS reads color phases on per sample basis for each slot.</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p style="text-align: center; color: blue; margin: 0;">Programming Notes</p> <p>This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.</p> <p>When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.</p> <p>When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).</p> <p>When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.</p> </div>	
12	Reserved	Format:	MBZ
11	Slot Group Select	Format:	MDC_RT_SGS
		This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.	
10:9	Reserved	Format:	MBZ
8	Render Target Message Subtype	Default Value:	1h
		Format:	Opcode
		SIMD8 single source message. Use slots [7:0] for the pixel enables, X/Y addresses, and oMask.	
		Programming Notes	
		The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].	
7:0	Binding Table Index	Format:	MDC_BTS
		Specifies the Binding Table Index for the message	

SIMD8 Render Target Write MSD

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD				
Source:		EuSubFunctionRenderDataPort		
Length Bias:		1		
DWord	Bit	Description		
0	31	Reserved Format: MBZ		
	30	Message Precision Subtype Default Value: 0h Format: Opcode Full precision data message		
	29	Reserved Format: MBZ		
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.		
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.		
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.		
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.</td> </tr> </tbody> </table>	Programming Notes	This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	Programming Notes			
	This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.			
	17:14	Message Type Default Value: 0Ch		

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD

	Format:	Opcode
	Render Target Write message	
13	Per-Sample PS Enable	
	Format:	Enable
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.	
	Programming Notes	
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.	
	When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.	
	When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).	
	When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.	
12	Last Render Target Select	
	Format:	Enable
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.	
11	Slot Group Select	
	Format:	MDC_RT_SGS
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.	
10:8	Render Target Message Subtype	
	Default Value:	4h
	Format:	Opcode
	SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask.	
	Programming Notes	
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].	
7:0	Binding Table Index	
	Format:	MDC_BTS
	Specifies the Binding Table Index for the message	

SIMD16 Render Target Read MSD

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD		
Source:		EuSubFunctionRenderDataPort
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Message Precision Subtype Default Value: 0h Format: Opcode Full precision data message Programming Notes This field must be programmed to 0
	29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	Reserved Format: MBZ
	17:14	Message Type Default Value: 0Dh Format: Opcode Render Target Read message
	13	Per-Sample PS Enable Format: Enable

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD

		<p>If set, PS reads color phases on per sample basis for each slot.</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p style="text-align: center; color: blue; margin: 0;">Programming Notes</p> <p>This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.</p> <p>When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.</p> <p>When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).</p> <p>When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.</p> </div>	
12	Reserved	Format:	MBZ
11	Slot Group Select	Format:	MDC_RT_SGS
		This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.	
10:9	Reserved	Format:	MBZ
8	Render Target Message Subtype	Default Value:	0h
		Format:	Opcode
		SIMD16 single source message. Use slots [15:0] for the pixel enables, X/Y addresses, and oMask.	
		<div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p style="text-align: center; color: blue; margin: 0;">Programming Notes</p> <p>The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].</p> </div>	
7:0	Binding Table Index	Format:	MDC_BTS
		Specifies the Binding Table Index for the message	

SIMD16 Render Target Write MSD

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD				
Source:		EuSubFunctionRenderDataPort		
Length Bias:		1		
DWord	Bit	Description		
0	31	Reserved Format: MBZ		
	30	Message Precision Subtype Default Value: 0h Format: Opcode Full precision data message		
	29	Reserved Format: MBZ		
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.		
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.		
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the 2-register header.		
	18	Per-Coarse Pixel PS outputs enable Format: Enable This bit indicates the render target write is a coarse pixel write. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td> This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor. When this bit is set and the message has oMask present, oMask represents the pixel enables within the Coarse Pixel in the row major order. </td> </tr> </tbody> </table>	Programming Notes	This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor. When this bit is set and the message has oMask present, oMask represents the pixel enables within the Coarse Pixel in the row major order.
	Programming Notes			
	This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor. When this bit is set and the message has oMask present, oMask represents the pixel enables within the Coarse Pixel in the row major order.			
	17:14	Message Type		

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

		Default Value:	0Ch
		Format:	Opcode
	Render Target Write message		
13	Per-Sample PS Enable		
	Format:	Enable	
	If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot.		
	Programming Notes		
	This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE.		
	When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0.		
	When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples).		
	When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.		
12	Last Render Target Select		
	Format:	Enable	
	This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.		
11	Slot Group Select		
	Format:	MDC_RT_SGS	
	This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.		
10:8	Render Target Message Subtype		
	Default Value:	0h	
	Format:	Opcode	
	SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask.		
	Programming Notes		
	The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].		
7:0	Binding Table Index		

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

	Format:	MDC_BTS
	Specifies the Binding Table Index for the message	

STATE_BASE_ADDRESS

STATE_BASE_ADDRESS			
Source:	BSpec		
Length Bias:	2		
<p>The STATE_BASE_ADDRESS command sets the base pointers for subsequent state, instruction, and media indirect object accesses by the GPE.</p> <p>For more information see the Base Address Utilization table in the Memory Access Indirection narrative topic.</p>			
Programming Notes			
<p>The following commands must be reissued following any change to the base addresses:</p> <ul style="list-style-type: none"> • 3DSTATE_CC_POINTERS • 3DSTATE_BINDING_TABLE_POINTERS • 3DSTATE_SAMPLER_STATE_POINTERS • 3DSTATE_VIEWPORT_STATE_POINTERS <p>Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance.</p> <p>If 3DSTATE_PS_EXTRA::Pixel Shader Is Per Coarse Pixel == 1, the 3DSTATE_CPS_POINTERS command must be reissued following any change to the dynamic state base address.</p> <p>SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set before programming STATE_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies in certain workload programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be traded off against performance by not implementing this programming note.</p> <p>SW must always program PIPE_CONTROL with Command Cache Invalidate Enable following programming of STATE_BASE_ADDRESS command when State Cache redirect to CS Section enable bit is set in MMIO register SLICE_COMMON_ECO_CHICKEN1 (0731Ch).</p> <p>SW must always program PIPE_CONTROL command with HDC Pipeline Flush set prior to programming of STATE_BASE_ADDRESS command for GPGPU/Media workloads i.e when pipeline select is GPGPU or Media via PIPELINE_SELECT command. This is required to ensure the write data out of the prior thread group are flushed out prior to the state changes due to the programming of STATE_BASE_ADDRESS command take place.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	0h GFXPIPE_COMMON
		Format:	OpCode
	26:24	3D Command Opcode	
		Default Value:	1h GFXPIPE_NONPIPELINED

STATE_BASE_ADDRESS																							
	<table border="1"> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Format:	OpCode																				
Format:	OpCode																						
23:16	3D Command Sub Opcode <table border="1"> <tr> <td>Default Value:</td> <td>01h STATE_BASE_ADDRESS</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	01h STATE_BASE_ADDRESS	Format:	OpCode																		
Default Value:	01h STATE_BASE_ADDRESS																						
Format:	OpCode																						
15:8	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																				
Format:	MBZ																						
7:0	DWord Length <table border="1"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>14h</td> <td>DWORD_COUNT_n [Default]</td> <td>Excludes DWord (0,1)</td> </tr> </tbody> </table>	Format:	=n	Value	Name	Description	14h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)														
Format:	=n																						
Value	Name	Description																					
14h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)																					
1..2	<table border="1"> <tr> <td>63:12</td> <td> General State Base Address <table border="1"> <tr> <td>Format:</td> <td>GA63_12</td> </tr> </table> <table border="1"> <thead> <tr> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is $\leq 2^{47}$. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Restriction</th> </tr> </thead> <tbody> <tr> <td>General State Base Address [47:12] + General State Buffer Size [31:12] must be $< 2^{48}$. It is illegal programming for this to be $\geq 2^{48}$. When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be $< 2^{48}$. It is illegal for this to be $\geq 2^{48}$.</td> </tr> </tbody> </table> </td> </tr> <tr> <td>11</td> <td> Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> </td> </tr> <tr> <td>10:4</td> <td> General State Memory Object Control State <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for indirect state using the General State Base Address, with the exception of the stateless data port accesses.</p> </td> </tr> <tr> <td>3:1</td> <td> Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> </td> </tr> </table>	63:12	General State Base Address <table border="1"> <tr> <td>Format:</td> <td>GA63_12</td> </tr> </table> <table border="1"> <thead> <tr> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is $\leq 2^{47}$. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Restriction</th> </tr> </thead> <tbody> <tr> <td>General State Base Address [47:12] + General State Buffer Size [31:12] must be $< 2^{48}$. It is illegal programming for this to be $\geq 2^{48}$. When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be $< 2^{48}$. It is illegal for this to be $\geq 2^{48}$.</td> </tr> </tbody> </table>	Format:	GA63_12	Description	Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].	Programming Notes	Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is $\leq 2^{47}$. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.	Restriction	General State Base Address [47:12] + General State Buffer Size [31:12] must be $< 2^{48}$. It is illegal programming for this to be $\geq 2^{48}$. When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be $< 2^{48}$. It is illegal for this to be $\geq 2^{48}$.	11	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	10:4	General State Memory Object Control State <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for indirect state using the General State Base Address, with the exception of the stateless data port accesses.</p>	Format:	MEMORY_OBJECT_CONTROL_STATE	3:1	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
63:12	General State Base Address <table border="1"> <tr> <td>Format:</td> <td>GA63_12</td> </tr> </table> <table border="1"> <thead> <tr> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Programming Notes</th> </tr> </thead> <tbody> <tr> <td>Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is $\leq 2^{47}$. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Restriction</th> </tr> </thead> <tbody> <tr> <td>General State Base Address [47:12] + General State Buffer Size [31:12] must be $< 2^{48}$. It is illegal programming for this to be $\geq 2^{48}$. When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be $< 2^{48}$. It is illegal for this to be $\geq 2^{48}$.</td> </tr> </tbody> </table>	Format:	GA63_12	Description	Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].	Programming Notes	Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is $\leq 2^{47}$. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.	Restriction	General State Base Address [47:12] + General State Buffer Size [31:12] must be $< 2^{48}$. It is illegal programming for this to be $\geq 2^{48}$. When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be $< 2^{48}$. It is illegal for this to be $\geq 2^{48}$.														
Format:	GA63_12																						
Description																							
Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].																							
Programming Notes																							
Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages. Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is $\leq 2^{47}$. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.																							
Restriction																							
General State Base Address [47:12] + General State Buffer Size [31:12] must be $< 2^{48}$. It is illegal programming for this to be $\geq 2^{48}$. When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be $< 2^{48}$. It is illegal for this to be $\geq 2^{48}$.																							
11	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																				
Format:	MBZ																						
10:4	General State Memory Object Control State <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for indirect state using the General State Base Address, with the exception of the stateless data port accesses.</p>	Format:	MEMORY_OBJECT_CONTROL_STATE																				
Format:	MEMORY_OBJECT_CONTROL_STATE																						
3:1	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ																				
Format:	MBZ																						

		STATE_BASE_ADDRESS		
	0	General State Base Address Modify Enable		
		Format:	Enable	
		The other fields in this DWord and the following DWord are updated only when this bit is set.		
		Value	Name	Description
		0h	Disable	Ignore the updated address.
		1h	Enable	Modify the address.
3	31:26	Reserved		
		Format:	MBZ	
	25:23	Reserved		
		Format:	MBZ	
	22:16	Stateless Data Port Access Memory Object Control State		
		Format:	MEMORY_OBJECT_CONTROL_STATE	
		Specifies the memory object control state for stateless data port accesses.		
15:13	Reserved			
	Format:	MBZ		
12:1	Reserved			
	Format:	MBZ		
0	Reserved			
	Format:	MBZ		
4.5	63:12	Surface State Base Address		
		Format:	GraphicsAddress[63:12]	
		Specifies the 4K-byte aligned base address for binding table and surface state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].		
	11	Reserved		
		Format:	MBZ	
10:4	Surface State Memory Object Control State			
	Format:	MEMORY_OBJECT_CONTROL_STATE		
	Specifies the memory object control state for indirect state using the Surface State Base Address .			
3:1	Reserved			
	Format:	MBZ		
0	Surface State Base Address Modify Enable			
	Format:	Enable		
	The other fields in this DWord and the following DWord are updated only when this bit is set.			

STATE_BASE_ADDRESS													
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated address.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the address.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Ignore the updated address.	1h	Enable	Modify the address.		
Value	Name	Description											
0h	Disable	Ignore the updated address.											
1h	Enable	Modify the address.											
		<p style="text-align: center;">Programming Notes</p> <p>Setting this bit to 1 in a batch buffer causes the resource streamer to stop; for performance reasons the SW should only place commands with this bit set in the ring buffer.</p> <p>Before programming the Surface State Base Address, the RS must be disabled. Within a batch buffer where the RS is enabled, RS may be disabled thru a MI_RS_CONTROL command with Resource Streamer Control cleared prior to the STATE_BASE_ADDRESS with Surface State Base Address Modify Enable set and then re-enabled with another MI_RS_CONTROL with Resource Streamer Control set.</p>											
6..7	63:12	<p>Dynamic State Base Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[63:12]</td> </tr> </table> <p style="text-align: center;">Description</p> <p>Specifies the 4K-byte aligned base address for sampler and viewport state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</p>	Format:	GraphicsAddress[63:12]									
Format:	GraphicsAddress[63:12]												
	11	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	10:4	<p>Dynamic State Memory Object Control State</p> <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for indirect state using the Dynamic State Base Address. Push constants defined in 3DSTATE_CONSTANT_(VS GS PS) commands do not use this control state, although they can use the corresponding base address. The memory object control state for push constants is defined within the command.</p>	Format:	MEMORY_OBJECT_CONTROL_STATE									
Format:	MEMORY_OBJECT_CONTROL_STATE												
	3:1	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ												
	0	<p>Dynamic State Base Address Modify Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>The other fields in this DWord and the following DWord are updated only when this bit is set.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated address.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the address.</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	Description	0h	Disable	Ignore the updated address.	1h	Enable	Modify the address.
Format:	Enable												
Value	Name	Description											
0h	Disable	Ignore the updated address.											
1h	Enable	Modify the address.											
8..9	63:12	<p>Indirect Object Base Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[63:12]</td> </tr> </table> <p>Specifies the 4K-byte aligned base address for indirect object load in MEDIA_OBJECT command.</p>	Format:	GraphicsAddress[63:12]									
Format:	GraphicsAddress[63:12]												

STATE_BASE_ADDRESS										
	11	Reserved Format: MBZ								
	10:4	Indirect Object Memory Object Control State Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect objects using the Indirect Object Base Address .								
	3:1	Reserved Format: MBZ								
	0	Indirect Object Base Address Modify Enable Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated address.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the address.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Ignore the updated address.	1h	Enable
Value	Name	Description								
0h	Disable	Ignore the updated address.								
1h	Enable	Modify the address.								
10..11	63:12	Instruction Base Address Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for all EU instruction accesses. GraphicsAddress[63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].								
	11	Reserved Format: MBZ								
	10:4	Instruction Memory Object Control State Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for EU instructions using the Instruction Base Address .								
	3:1	Reserved Format: MBZ								
	0	Instruction Base Address Modify Enable Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated address.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the address.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Ignore the updated address.	1h	Enable
Value	Name	Description								
0h	Disable	Ignore the updated address.								
1h	Enable	Modify the address.								
12	31:12	General State Buffer Size Format: U20								

STATE_BASE_ADDRESS

		<p>FormatDesc</p> <p>This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.</p>									
	11:1	<p>Reserved</p> <p>Format: MBZ</p>									
	0	<p>General State Buffer Size Modify Enable</p> <p>Format: Enable</p> <p>The fields in this DWord are updated only when this bit is set.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated bound.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the updated bound.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Ignore the updated bound.	1h	Enable	Modify the updated bound.
Value	Name	Description									
0h	Disable	Ignore the updated bound.									
1h	Enable	Modify the updated bound.									
13	31:12	<p>Dynamic State Buffer Size</p> <p>Format: U20</p> <p>FormatDesc</p> <p>This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.</p>									
	11:1	<p>Reserved</p> <p>Format: MBZ</p>									
	0	<p>Dynamic State Buffer Size Modify Enable</p> <p>Format: Enable</p> <p>FormatDesc</p> <p>The fields in this DWord are updated only when this bit is set.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated bound.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the updated bound.</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Ignore the updated bound.	1h	Enable	Modify the updated bound.
Value	Name	Description									
0h	Disable	Ignore the updated bound.									
1h	Enable	Modify the updated bound.									
14	31:12	<p>Indirect Object Buffer Size</p> <p>Format: U20</p> <p>FormatDesc</p> <p>This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.</p>									
	11:1	<p>Reserved</p>									

STATE_BASE_ADDRESS													
	<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ										
Format:	MBZ												
0	<p>Indirect Object Buffer Size Modify Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>FormatDesc</p> <p>The fields in this DWord are updated only when this bit is set.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated bound.</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the updated bound.</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	Description	0h	Disable	Ignore the updated bound.	1h	Enable	Modify the updated bound.	
Format:	Enable												
Value	Name	Description											
0h	Disable	Ignore the updated bound.											
1h	Enable	Modify the updated bound.											
15	<p>31:12 Instruction Buffer Size</p> <table border="1"> <tr> <td>Format:</td> <td>U20</td> </tr> </table> <p>FormatDesc</p> <p>This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.</p> <p>11:1 Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> <p>0 Instruction Buffer size Modify Enable</p> <table border="1"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>FormatDesc</p> <p>The fields in this DWord are updated only when this bit is set.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated bound.</td> </tr> </tbody> </table>	Format:	U20	Format:	MBZ	Format:	Enable	Value	Name	Description	0h	Disable	Ignore the updated bound.
Format:	U20												
Format:	MBZ												
Format:	Enable												
Value	Name	Description											
0h	Disable	Ignore the updated bound.											
16..17	<p>63:12 Bindless Surface State Base Address</p> <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[63:12]</td> </tr> </table> <p>Specifies the 4K-byte aligned base address for bindless surface state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].</p> <p>11 Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table> <p>10:4 Bindless Surface State Memory Object Control State</p> <table border="1"> <tr> <td>Format:</td> <td>MEMORY_OBJECT_CONTROL_STATE</td> </tr> </table> <p>Specifies the memory object control state for indirect state using the Bindless Surface State Base Address.</p>	Format:	GraphicsAddress[63:12]	Format:	MBZ	Format:	MEMORY_OBJECT_CONTROL_STATE						
Format:	GraphicsAddress[63:12]												
Format:	MBZ												
Format:	MEMORY_OBJECT_CONTROL_STATE												

STATE_BASE_ADDRESS											
	3:1	Reserved Format: _____ MBZ									
	0	Bindless Surface State Base Address Modify Enable Format: _____ Enable <div style="text-align: center;">Description</div> The other fields in this DWord and the following two DWords are updated only when this bit is set. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 65%;">Description</th> </tr> </thead> <tbody> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated address</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the address</td> </tr> </tbody> </table>	Value	Name	Description	0h	Disable	Ignore the updated address	1h	Enable	Modify the address
	Value	Name	Description								
	0h	Disable	Ignore the updated address								
	1h	Enable	Modify the address								
18	31:12	Bindless Surface State Size Format: _____ U20 This field indicates the size-1 of the Bindless Surface State buffer in 64-Byte increments. Any SSO beyond this maximum size points to offset 0. Example: If the buffer contains 512 surface states, then this field must be programmed to 0x1FF (511 decimal).									
	11:0	Reserved Format: _____ MBZ									
19..20	63:12	Bindless Sampler State Base Address Format: _____ GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for bindless sampler state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].									
	11	Reserved Format: _____ MBZ									
	10:4	Bindless Sampler State Memory Object Control State Format: _____ MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the Bindless Sampler State Base Address .									
	3:1	Reserved Format: _____ MBZ									
	0	Bindless Sampler State Base Address Modify Enable Format: _____ Enable The other fields in this DWord and the following two DWords are updated only when this bit is set. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 20%;">Name</th> <th style="width: 65%;">Description</th> </tr> </thead> <tbody> </tbody> </table>	Value	Name	Description						
Value	Name	Description									

STATE_BASE_ADDRESS								
		<table border="1"> <tr> <td>0h</td> <td>Disable</td> <td>Ignore the updated address</td> </tr> <tr> <td>1h</td> <td>Enable</td> <td>Modify the address</td> </tr> </table>	0h	Disable	Ignore the updated address	1h	Enable	Modify the address
0h	Disable	Ignore the updated address						
1h	Enable	Modify the address						
21	31:12	Bindless Sampler State Buffer Size Format: _____ U20 This field specifies the size of the buffer in 4K pages. Any access that goes beyond the end of the buffer (as defined by the Sampler State Buffer Size) will use an offset of 0.						
		Reserved Format: _____ MBZ						
	11:0							



STATE_COMPUTE_MODE

STATE_COMPUTE_MODE		
Source:	RenderCS, ComputeCS	
Length Bias:	2	
This is a non-pipeline state command and is a general compute programming state that can be shared from the top to bottom of the pipeline.		
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h GFXPIPE
		Format: OpCode
	28:27	Command SubType
		Default Value: 0h GFXPIPE_COMMON Format: OpCode
	26:24	3D Command Opcode
		Default Value: 1h GFXPIPE_NONPIPELINED Format: OpCode
23:16	3D Command Sub Opcode	
	Default Value: 05h STATE_COMPUTE_MODE Format: OpCode	
15:8	Reserved	
7:0	DWord Length	
	Default Value: 0h Excludes DWord (0,1)	
	Format: =n	
1	31:16	Mask
		Format: Enable[16] This field is the mask bits for the state bits below. This is a bit wise mask where the bit number-16 is the value of the corresponding bit being masked in the same data word. For example, if you want to update state for bits 3:2 then bits 19:18 must be set.
	15	Reserved
	14	Reserved
	13	Disable L1 Invalidate for non-L1-cacheable Writes
	Format: Disable When this bit is set, HDC global memory write requests that are marked "non-L1-cacheable"	

STATE_COMPUTE_MODE

	<p>(either due to MOCS setting or L1-cache-disable modebits set in this register) will not send "Invalidation" request to the SamplerL1 cache. The implication of this bit being set is that HDC will not maintain RAW and WAR ordering between L1-cacheable and non-L1-cacheable requests to the same address. URB writes never sends Invalidate commands to Sampler L1 cache (regardless of this bit).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Enable [Default]</td> <td>HDC Non-L1-cacheable writes to Global memory will send Invalidate command to Sampler L1 cache.</td> </tr> <tr> <td>1b</td> <td>Disable</td> <td>HDCNon-L1-cacheable writes to Global memory will NOT send Invalidate command to Sampler L1 cache.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Enable [Default]	HDC Non-L1-cacheable writes to Global memory will send Invalidate command to Sampler L1 cache.	1b	Disable	HDCNon-L1-cacheable writes to Global memory will NOT send Invalidate command to Sampler L1 cache.							
Value	Name	Description															
0b	Enable [Default]	HDC Non-L1-cacheable writes to Global memory will send Invalidate command to Sampler L1 cache.															
1b	Disable	HDCNon-L1-cacheable writes to Global memory will NOT send Invalidate command to Sampler L1 cache.															
12:11	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ														
Format:	MBZ																
10	<p>Binding Table Alignment</p> <p>When this bit is set, the format of the compute context's binding table is SW binding table format whether Binding Table Pool is enabled or disabled.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Legacy [Default]</td> <td>Binding table pointer 15:5 Maps to 15:5 for INTERFACE_DESCRIPTOR DATA. Binding table pointer 15:5 Maps to 15:5 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is disabled. Binding table pointer 15:5 Maps to 16:6 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is enabled.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td colspan="2">Compute dispatches from the RenderCS context use GT_MODE[10] and not this bit. This bit is only used by ComputeCS contexts.</td> </tr> </tbody> </table>	Value	Name	Description	0	Legacy [Default]	Binding table pointer 15:5 Maps to 15:5 for INTERFACE_DESCRIPTOR DATA. Binding table pointer 15:5 Maps to 15:5 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is disabled. Binding table pointer 15:5 Maps to 16:6 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is enabled.	Programming Notes		Compute dispatches from the RenderCS context use GT_MODE[10] and not this bit. This bit is only used by ComputeCS contexts.							
Value	Name	Description															
0	Legacy [Default]	Binding table pointer 15:5 Maps to 15:5 for INTERFACE_DESCRIPTOR DATA. Binding table pointer 15:5 Maps to 15:5 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is disabled. Binding table pointer 15:5 Maps to 16:6 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is enabled.															
Programming Notes																	
Compute dispatches from the RenderCS context use GT_MODE[10] and not this bit. This bit is only used by ComputeCS contexts.																	
9:7	<p>Async Compute Thread Limit</p> <p>Specifies the maximum number of active Compute CS threads to run in a DSS when the 3D Pipe is active and a Z-pass is not running. When the 3D Pipe is not active or when a Z-pass is running, the maximum number of active Compute CS threads is specified by Maximum Number of Threads in CFE_STATE command.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> <th>Programming Notes</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disabled [Default]</td> <td>No limit applied. Maximum Number of Threads is the only limit on Compute CS threads.</td> <td></td> </tr> <tr> <td>1</td> <td>Max 2</td> <td>Maximum of 2 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to about 1 thread per EU row.</td> <td></td> </tr> <tr> <td>2</td> <td>Max 8</td> <td>Maximum of 8 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to</td> <td>This value is the recommended SW setting, to balance forward progress on Async Compute and</td> </tr> </tbody> </table>	Value	Name	Description	Programming Notes	0	Disabled [Default]	No limit applied. Maximum Number of Threads is the only limit on Compute CS threads.		1	Max 2	Maximum of 2 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to about 1 thread per EU row.		2	Max 8	Maximum of 8 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to	This value is the recommended SW setting, to balance forward progress on Async Compute and
Value	Name	Description	Programming Notes														
0	Disabled [Default]	No limit applied. Maximum Number of Threads is the only limit on Compute CS threads.															
1	Max 2	Maximum of 2 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to about 1 thread per EU row.															
2	Max 8	Maximum of 8 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to	This value is the recommended SW setting, to balance forward progress on Async Compute and														

STATE_COMPUTE_MODE

		about 1 thread per EU .	3D Pipe dispatches.
3	Max 16	Maximum of 16 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to about 2 threads per EU .	
4	Max 20	Maximum of 20 Fused-EU threads per DSS, when 3D Pipe is active.	
5	Max 24	Maximum of 24 Fused-EU threads per DSS, when 3D Pipe is active.	
6	Max 28	Maximum of 28 Fused-EU threads per DSS, when 3D Pipe is active.	
7	Max 32	Maximum of 32 Fused-EU threads per DSS, when 3D Pipe is active. This sets the Async Compute thread limit to about half the threads per EU .	
6:5	Reserved		
	Format:	MBZ	
4:3	Force Non-Coherent		
	Format:	U2	
	Force all Data Cache Data Port access to be Non-Coherent (virtual addresses) and non-faultable regardless of the surface state or binding table index.		
	Value	Name	Description
	2h	Force GPU Non-Coherent	GPU accesses are forced as non-coherent with other GPU, as well as with the CPU.
	Programming Notes		
	Only change this mode after a pipeflush and cache flush (all threads and their all accesses completed).		
	CPU-GPU or GPU-GPU coherency is not supported. Hence the driver must set this field to 0x2 (Force GPU non-coherent), if the data-port message has coherency enabled via BTI or surface state.		
2:0	Reserved		
	Format:	MBZ	

STATE_SIP

STATE_SIP			
Source:	BSpec		
Length Bias:	2		
The STATE_SIP command specifies the starting instruction location of the System Routine that is shared by all threads in execution.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	Command SubType	
		Default Value:	0h GFXPIPE_COMMON
	26:24	3D Command Opcode	
		Default Value:	1h GFXPIPE_NONPIPELINED
23:16	3D Command Sub Opcode		
	Default Value:	02h STATE_SIP	
15:8	Reserved		
7:0	Format:	MBZ	
	DWord Length		
	Format:	=n	
1.2	63:4	System Instruction Pointer	
		Format:	InstructionBaseOffset[63:4]
		Specifies the instruction address of the system routine associated with the current context as a 128-bit granular offset from the Instruction Base Address. SIP is shared by all threads in execution. The address specifies the double quadword aligned instruction location. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].	
	Programming Notes		
		This portion of the command is not context save/restored. The context image may restore this command as a 2 dword command rather than a 3 dword command.	
3:0	Reserved		



STATE_SIP	
Format:	MBZ

Subtraction with Borrow

subb - Subtraction with Borrow			
Source:	Eulsa		
Length Bias:	4		
<p>The subb instruction performs component-wise subtraction of src0 and src1 and stores the results in dst, it also stores the borrow into acc. If the operation produces a borrow (src0 < src1), write 0x00000001 to acc, else write 0x00000000 to acc.</p>			
Format:	<code>[(pred)] subb[.cmod] (exec_size) dst src0 src1</code>		
Restriction			
AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.			
Syntax			
<code>[(pred)] subb[.cmod] (exec_size) reg reg reg [(pred)] subb[.cmod] (exec_size) reg reg imm32</code>			
Pseudocode			
<pre>Evaluate(WrEn); for (n = 0; n < exec_size; n++) { if (WrEn.chan[n]) { dst.chan[n] = src0.chan[n] - src1.chan[n]; acc.chan[n] = borrow(src.chan[n] - src1.chan[n]); } }</pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	Y	Y	N
Src Types	Dst Types		
UD	UD		
DWord	Bit	Description	
0..3	127:126	Reserved	
		Exists If:	<code>([Src1.IsImm]==false)</code>
	Format:	MBZ	
	127:96	Src1.ImmValue[31:0]	
Exists If:	<code>([Src1.IsImm]==true)</code>		
125:122	Reserved		
	Exists If:	<code>([Src1.IsImm]==false)</code>	
Format:	MBZ		
121:120	Src1.Mod		
	Exists If:	<code>([Src1.IsImm]==false)</code>	
Format:	SrcMod		

subb - Subtraction with Borrow

119:116	Src1.VertStride		
	Exists If:	([Src1.IsImm]==false)	
	Format:	VertStride	
	115:113	Src1.Width	
		Exists If:	([Src1.IsImm]==false)
	Format:	Width	
	112	Src1.AddrMode	
		Exists If:	([Src1.IsImm]==false)
	Format:	AddrMode	
	111:98	Src1.Operand	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	DirectOperand	
	111:98	Src1.Operand	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	IndirectOperand	
	97:96	Src1.HorzStride	
Exists If:		([Src1.IsImm]==false)	
Format:	HorzStride		
95:92	CondCtrl		
	Format:	FlagModifier	
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==false)	
Format:	RegDataType		
91:88	Src1.DataType		
	Exists If:	([Src1.IsImm]==true)	
Format:	ImmDataType		
87:84	Src0.VertStride		
	Format:	VertStride	
83:81	Src0.Width		
	Format:	Width	
80	Src0.AddrMode		
	Format:	AddrMode	
79:66	Src0.Operand		
	Exists If:	([Src0.AddrMode]==Direct)	
Format:	DirectOperand		
79:66	Src0.Operand		

subb - Subtraction with Borrow											
	<table border="1"> <tr> <td>Exists If:</td> <td>([Src0.AddrMode] == Indirect)</td> </tr> <tr> <td>Format:</td> <td>IndirectOperand</td> </tr> </table>	Exists If:	([Src0.AddrMode] == Indirect)	Format:	IndirectOperand						
Exists If:	([Src0.AddrMode] == Indirect)										
Format:	IndirectOperand										
65:64	<table border="1"> <tr> <td colspan="2">Src0.HorzStride</td> </tr> <tr> <td>Format:</td> <td>HorzStride</td> </tr> </table>	Src0.HorzStride		Format:	HorzStride						
Src0.HorzStride											
Format:	HorzStride										
63:50	<table border="1"> <tr> <td colspan="2">Dst.Operand</td> </tr> <tr> <td>Exists If:</td> <td>(([Dst.AddrMode] == Direct)</td> </tr> <tr> <td>Format:</td> <td>DirectOperand</td> </tr> </table>	Dst.Operand		Exists If:	(([Dst.AddrMode] == Direct)	Format:	DirectOperand				
Dst.Operand											
Exists If:	(([Dst.AddrMode] == Direct)										
Format:	DirectOperand										
63:50	<table border="1"> <tr> <td colspan="2">Dst.Operand</td> </tr> <tr> <td>Exists If:</td> <td>(([Dst.AddrMode] == Indirect)</td> </tr> <tr> <td>Format:</td> <td>IndirectOperand</td> </tr> </table>	Dst.Operand		Exists If:	(([Dst.AddrMode] == Indirect)	Format:	IndirectOperand				
Dst.Operand											
Exists If:	(([Dst.AddrMode] == Indirect)										
Format:	IndirectOperand										
49:48	<table border="1"> <tr> <td colspan="2">Dst.HorzStride</td> </tr> <tr> <td>Format:</td> <td>HorzStride</td> </tr> </table>	Dst.HorzStride		Format:	HorzStride						
Dst.HorzStride											
Format:	HorzStride										
47	<table border="1"> <tr> <td colspan="2">Src1.IsImm</td> </tr> <tr> <td colspan="2">This field indicate that Source 1 operand is carrying an immediate value.</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td style="text-align: center;">0</td> <td>false [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </table>	Src1.IsImm		This field indicate that Source 1 operand is carrying an immediate value.		Value	Name	0	false [Default]	1	true
Src1.IsImm											
This field indicate that Source 1 operand is carrying an immediate value.											
Value	Name										
0	false [Default]										
1	true										
46	<table border="1"> <tr> <td colspan="2">Src0.IsImm</td> </tr> <tr> <td colspan="2">This field indicate that Source 0 operand is carrying an immediate value.</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td style="text-align: center;">0</td> <td>false [Default]</td> </tr> <tr> <td style="text-align: center;">1</td> <td>true</td> </tr> </table>	Src0.IsImm		This field indicate that Source 0 operand is carrying an immediate value.		Value	Name	0	false [Default]	1	true
Src0.IsImm											
This field indicate that Source 0 operand is carrying an immediate value.											
Value	Name										
0	false [Default]										
1	true										
45:44	<table border="1"> <tr> <td colspan="2">Src0.Mod</td> </tr> <tr> <td>Format:</td> <td>SrcMod</td> </tr> </table>	Src0.Mod		Format:	SrcMod						
Src0.Mod											
Format:	SrcMod										
43:40	<table border="1"> <tr> <td colspan="2">Src0.DataType</td> </tr> <tr> <td>Exists If:</td> <td>(([Src0.IsImm] == false)</td> </tr> <tr> <td>Format:</td> <td>RegDataType</td> </tr> </table>	Src0.DataType		Exists If:	(([Src0.IsImm] == false)	Format:	RegDataType				
Src0.DataType											
Exists If:	(([Src0.IsImm] == false)										
Format:	RegDataType										
43:40	<table border="1"> <tr> <td colspan="2">Src0.DataType</td> </tr> <tr> <td>Exists If:</td> <td>(([Src0.IsImm] == true)</td> </tr> <tr> <td>Format:</td> <td>ImmDataType</td> </tr> </table>	Src0.DataType		Exists If:	(([Src0.IsImm] == true)	Format:	ImmDataType				
Src0.DataType											
Exists If:	(([Src0.IsImm] == true)										
Format:	ImmDataType										
39:36	<table border="1"> <tr> <td colspan="2">Dst.DataType</td> </tr> <tr> <td>Format:</td> <td>RegDataType</td> </tr> </table>	Dst.DataType		Format:	RegDataType						
Dst.DataType											
Format:	RegDataType										
35	<table border="1"> <tr> <td colspan="2">Dst.AddrMode</td> </tr> <tr> <td>Format:</td> <td>AddrMode</td> </tr> </table>	Dst.AddrMode		Format:	AddrMode						
Dst.AddrMode											
Format:	AddrMode										
34	<table border="1"> <tr> <td colspan="2">Saturate</td> </tr> <tr> <td>Format:</td> <td>Saturate</td> </tr> </table>	Saturate		Format:	Saturate						
Saturate											
Format:	Saturate										

subb - Subtraction with Borrow											
33	AccWrCtrl Format: AccWrCtrl										
32	AtomicCtrl Format: AtomicCtrl										
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Normal [Default]</td> <td>Normal. Per channel write enable used for final write enable generation.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>NoMask</td> <td>NoMask. Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td> </tr> </tbody> </table>		Value	Name	Description	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.	1	NoMask	NoMask. Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
Value	Name	Description									
0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.									
1	NoMask	NoMask. Skips the check for PciP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.									
30	Reserved										
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>		Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description									
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.									
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.									
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>		Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description									
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.									
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.									
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the										

subb - Subtraction with Borrow			
	content of the selected flag register.		
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>		
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Synchronize

sync - Synchronize	
Source:	Eulsa
Length Bias:	4
<p>Wait on Dependency performs various operations related to synchronization such as waiting on registers (barriers registers) or for software scoreboarding (SWSB), which is used to specify pipeline hazards to the EU. The instruction has several sub-operations (function controls), including:</p> <ul style="list-style-type: none"> • nop (0000b): no operation (encoded SWSB information available to every instruction is still honored). This might be used if an instruction depends on two different out-of-order sources. The consumer can only specify a dependency on one, hence an extra instruction must be added for this. • Reserved (0001b): reserved for future expansion. • allrd (0010b): blocks until all out-of-order sources are read (e.g. input arguments to a send or math op). • allwr (0011b): blocks until all out-of-order destinations are written back (e.g. writes from a send or math op). • Reserved (0100-1100b): reserved for future expansion • fence (1101b): blocks on the notification register for fence response. When fence response is received from message gateway, bit 0 of n2 notification register is set. Instruction sync.fence blocks until the bit is set and clears before progressing to the next instruction. • bar (1110b): blocks on the notification register for barriers response. When barrier response is received from message gatewaybits corresponding to the barrier id are set in the notification register n0. Instruction sync.bar(barrier id) blocks until the bit corresponding to the barrier id is set, and clears it before progressing to the next instruction. • host (1111b): blocks on the notification register for host interaction. When host notification is received, the bit 0 of n1 notification register is set. Instruction sync.host blocks until the bit is set and clears it before progressing to the next instruction. <p>See the SyncFC BXML enum for more information.</p>	
Format: sync.[sync_fc] src0	
Programming Notes	
The format is that of a basic instruction. The immediate operand is encoded as src0 and may explicitly be null if not used. Src1 and dst must be null.	
Syntax	
<pre>sync.nop null [instopts] sync.allrd (null imm32) [instopts] sync.allwr (null imm32) [instopts] sync.bar null[instopts] sync.host null [instopts]</pre>	
Pseudocode	

sync - Synchronize

```

Evaluate(WrEn);
switch (func) {
case nop:
    // regular SWSB dep check from instruction options executes
    break;
case allrd:
    for (sbid = 0; sbid < MAX_SBIDS; sbid++) {
        if (Src0.IsImm) {
            // wait until selected OOO reads are finished
            if(Src0.ImmValue[sbid]) wait_on_sbid_read_access(sbid); // transition to
wait_dst or idle
        } else {
            // wait until all OOO reads are finished
            wait_on_sbid_read_access(sbid); // transition to wait_dst or idle
        }
    }
    break;
case allwr:
    for (sbid = 0; sbid < MAX_SBIDS; sbid++) {
        if (Src0.IsImm) {
            // wait until selected OOO writes are finished
            if(Src0.ImmValue[sbid]) wait_on_sbid_write_access(sbid); // transition
to idle
        } else {
            // wait until selected OOO writes are finished
            wait_on_sbid_write_access(sbid); // transition to idle
        }
    }
    break;
case bar:
    wait_on_barrier_notification(1 << Src0);
    if (Src0.IsImm) {
        wait_on_barrier_notification(1 << Src0.ImmValue[4:0]) // waits until the
corresponding barrier bit is set
    } else if (Src0.RegFile == ARF) {
        wait_on_barrier_notification(1 << Src0[4:0]) // waits until the
corresponding barrier bit is set
    } else {
        wait_on_barrier_notification(1) // waits until the barrier bit 0 is set
    }
    break;
case host:
    wait_on_host_notification(); // waits until the host signals the host barrier
    break;
}

```

Predication	Conditional Modifier	Saturation	Source Modifier	Subfunctions
N	N	N	N	Unspecified

Src Types

*B,*W,*D,*Q, HF, F, DF

DWord	Bit	Description				
0..3	127:96	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Exists If:</td> <td>([Src0.IsImm]==false)</td> </tr> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Exists If:	([Src0.IsImm]==false)	Format:	MBZ
Exists If:	([Src0.IsImm]==false)					
Format:	MBZ					

sync - Synchronize

127:96	Src0.ImmValue[31:0]		
	Exists If:	([Src0.IsImm]==true)	
	95:92	SyncCtrl	
		Format:	SyncFC
	91:88	Reserved	
		Format:	MBZ
	87	Reserved	
		Format:	MBZ
	86:80	Reserved	
		Format:	MBZ
	79:66	Reserved	
		Format:	MBZ
	65:50	Reserved	
		Format:	MBZ
	49:48	Dst.HorzStride	
		Value	Name
		01b	1 elements [Default]
		Others	Reserved
	47	Reserved	
		Format:	MBZ
46	Src0.IsImm		
	This field indicate that Source 0 operand is carrying an immediate value.		
	Value	Name	
	0	false	
1	true		
45:44	Reserved		
	Format:	MBZ	
43:40	Reserved		
	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
43:40	Src0.DataType		
	Exists If:	([Src0.IsImm]==true)	
	Format:	ImmDataType	
39:33	Reserved		
	Format:	MBZ	
32	AtomicCtrl		

sync - Synchronize

sync - Synchronize		
	Format:	AtomicCtrl
31	MaskCtrl Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name
	0	Normal [Default]
	1	NoMask
	Description	
	Normal. Per channel write enable used for final write enable generation.	
	NoMask.Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
30	Reserved	
29	CmptCtrl Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.	
	Value	Name
	0	NoCompaction [Default]
	1	Compacted
	Description	
	No compaction. 128-bit native instruction supporting all instruction options.	
	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.	
28	PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields	
	Value	Name
	0	Positive [Default]
	1	Negative
	Description	
	Positive polarity of predication. Use the predication mask produced by PredCtrl.	
	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.	
27:24	PredCtrl Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.	
23	FlagRegNum[0]	

sync - Synchronize			
	This field specifies bit[0] of the register number for a flag register operand.		
22	<p>FlagSubRegNum</p> <p>This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>		
21:19	<p>ChanOff</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff
Format:	ChanOff		
18:16	<p>ExecSize</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">ExecSize</td> </tr> </table> <p>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</p>	Format:	ExecSize
Format:	ExecSize		
15:0	<p>Header</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%; text-align: right;">Header</td> </tr> </table>	Format:	Header
Format:	Header		

Typed Surface Read MSD

MSD1R_TS - Typed Surface Read MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type Default Value: 05h

MSD1R_TS - Typed Surface Read MSD					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Opcode</td> </tr> <tr> <td colspan="2">Typed Surface Read message</td> </tr> </table>	Format:	Opcode	Typed Surface Read message	
Format:	Opcode				
Typed Surface Read message					
13:12	<p>Slot Group</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MDC_SG3</td> </tr> <tr> <td colspan="2">Specifies the Slot Group mode of the message (which slots are processed)</td> </tr> </table>	Format:	MDC_SG3	Specifies the Slot Group mode of the message (which slots are processed)	
Format:	MDC_SG3				
Specifies the Slot Group mode of the message (which slots are processed)					
11:8	<p>Channel Mask</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MDC_CMASK</td> </tr> <tr> <td colspan="2">Specifies which RGBA channels are included in the message payload.</td> </tr> </table>	Format:	MDC_CMASK	Specifies which RGBA channels are included in the message payload.	
Format:	MDC_CMASK				
Specifies which RGBA channels are included in the message payload.					
7:0	<p>Binding Table Index</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MDC_BTS</td> </tr> <tr> <td colspan="2">Specifies the Binding Table Index for the message</td> </tr> </table>	Format:	MDC_BTS	Specifies the Binding Table Index for the message	
Format:	MDC_BTS				
Specifies the Binding Table Index for the message					

Typed Surface Write MSD

MSD1W_TS - Typed Surface Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type Default Value: 0Dh

MSD1W_TS - Typed Surface Write MSD

		Format:	Opcode
		Typed Surface Write message	
13:12	Slot Group	Format:	MDC_SG3
		Specifies the Slot Group mode of the message (which slots are processed)	
11:8	Channel Mask	Format:	MDC_CMASK
		Specifies which RGBA channels are included in the message payload.	
7:0	Binding Table Index	Format:	MDC_BTS
		Specifies the Binding Table Index for the message	

Untyped Surface Read MSD

MSD1R_US - Untyped Surface Read MSD		
Source:	EuSubFunctionDataPort1	
Length Bias:	1	
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type Default Value: 01h

MSD1R_US - Untyped Surface Read MSD					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Opcode</td> </tr> <tr> <td colspan="2">Untyped Surface Read message</td> </tr> </table>	Format:	Opcode	Untyped Surface Read message	
Format:	Opcode				
Untyped Surface Read message					
13:12	<p>SIMD Mode</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MDC_SM3</td> </tr> <tr> <td colspan="2">Specifies the SIMD mode of the message (number of slots processed)</td> </tr> </table>	Format:	MDC_SM3	Specifies the SIMD mode of the message (number of slots processed)	
Format:	MDC_SM3				
Specifies the SIMD mode of the message (number of slots processed)					
11:8	<p>Channel Mask</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MDC_CMASK</td> </tr> <tr> <td colspan="2">Specifies which RGBA channels are included in the message payload.</td> </tr> </table>	Format:	MDC_CMASK	Specifies which RGBA channels are included in the message payload.	
Format:	MDC_CMASK				
Specifies which RGBA channels are included in the message payload.					
7:0	<p>Binding Table Index</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MDC_BTS_SLM_A32</td> </tr> <tr> <td colspan="2">Specifies the Binding Table Index for the message</td> </tr> </table>	Format:	MDC_BTS_SLM_A32	Specifies the Binding Table Index for the message	
Format:	MDC_BTS_SLM_A32				
Specifies the Binding Table Index for the message					

Untyped Surface Write MSD

MSD1W_US - Untyped Surface Write MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type Default Value: 09h

MSD1W_US - Untyped Surface Write MSD					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>Opcode</td> </tr> <tr> <td colspan="2">Untyped Surface Write message</td> </tr> </table>	Format:	Opcode	Untyped Surface Write message	
Format:	Opcode				
Untyped Surface Write message					
13:12	<p>SIMD Mode</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MDC_SM3</td> </tr> <tr> <td colspan="2">Specifies the SIMD mode of the message (number of slots processed)</td> </tr> </table>	Format:	MDC_SM3	Specifies the SIMD mode of the message (number of slots processed)	
Format:	MDC_SM3				
Specifies the SIMD mode of the message (number of slots processed)					
11:8	<p>Channel Mask</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MDC_UW_CMASK</td> </tr> <tr> <td colspan="2">Specifies which RGBA channels are included in the message payload.</td> </tr> </table>	Format:	MDC_UW_CMASK	Specifies which RGBA channels are included in the message payload.	
Format:	MDC_UW_CMASK				
Specifies which RGBA channels are included in the message payload.					
7:0	<p>Binding Table Index</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MDC_BTS_SLM_A32</td> </tr> <tr> <td colspan="2">Specifies the Binding Table Index for the message</td> </tr> </table>	Format:	MDC_BTS_SLM_A32	Specifies the Binding Table Index for the message	
Format:	MDC_BTS_SLM_A32				
Specifies the Binding Table Index for the message					

URB Dword Read MSD

MSDUR_DWS - URB Dword Read MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	Reserved Format: MBZ
	17	Per Slot Offset Present Format: Enable Specifies if per-slot offset message payload is present.
	16	Reserved Format: MBZ
	15	Channel Mask Present Default Value: 0 Not Present Format: Opcode Must be clear on read messages, indicating the Channel Mask Message phase is not present.
	14:4	Global Offset Format: U11 Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If Per Slot Offset Present is set, this global offset is added to each of the slot offsets to form the

MSDUR_DWS - URB Dword Read MSD			
		overall offset.	
		Value	Name
		[0-2047]	
	3:0	URB Opcode	
		Format:	Opcode
		Value	Name
		Description	
		8	URB_SIMD8_READ [Default]
			SIMD8 URB Dword Read message. Reads 1..8 Dwords, based on RLEN.

URB Dword Write MSD

MSDUW_DWS - URB Dword Write MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: MBZ
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	Reserved Format: MBZ
	17	Per Slot Offset Present Format: Enable Specifies if per-slot offset message payload is present. If present, it will be added to the Global Offset .
	16	Reserved Format: MBZ
	15	Channel Mask Present Default Value: 0 Not Present Format: Opcode Indicates the channel Mask Message phase is not present.
	14:4	Global Offset Format: U11 Specifies the offset, in units of Oword elements, from the start of the URB handle for the access.

MSDUW_DWS - URB Dword Write MSD								
		If Per Slot Offset is set, the global offset is added to those offsets to form the overall offset.						
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>[0-2047]</td> <td></td> </tr> </tbody> </table>	Value	Name	[0-2047]			
Value	Name							
[0-2047]								
	3:0	URB Opcode						
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td style="width: 50%;">Opcode</td> </tr> </table>	Format:	Opcode				
Format:	Opcode							
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 10%; text-align: center;">Value</th> <th style="width: 30%; text-align: center;">Name</th> <th style="width: 60%; text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">7</td> <td>URB_SIMD8_WRITE [Default]</td> <td>SIMD8 URB Dword Write message. Writes 1..8 Dwords, based on RLEN and Channel Mask.</td> </tr> </tbody> </table>	Value	Name	Description	7	URB_SIMD8_WRITE [Default]	SIMD8 URB Dword Write message. Writes 1..8 Dwords, based on RLEN and Channel Mask.
Value	Name	Description						
7	URB_SIMD8_WRITE [Default]	SIMD8 URB Dword Write message. Writes 1..8 Dwords, based on RLEN and Channel Mask.						

URB Masked Dword Write MSD

MSDUW_MDWS - URB Masked Dword Write MSD		
Source:		EuSubFunctionReadOnlyDataPort
Length Bias:		1
DWord	Bit	Description
0	31:29	Reserved Format: _____ MBZ
	28:25	Message Length Format: _____ U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: _____ U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Default Value: _____ 1 Present Format: _____ Opcode Indicates that the message requires a header.
	18	Reserved Format: _____ MBZ
	17	Per Slot Offset Present Format: _____ Enable Specifies if per-slot offset message payload is present. If present, it will be added to the Global Offset .
	16	Reserved Format: _____ MBZ
	15	Channel Mask Present Default Value: _____ 1 Present Format: _____ Opcode Indicates the Channel Mask Message phase is present and will be used to mask which data elements written.
	14:4	Global Offset Format: _____ U11

MSDUW_MDWS - URB Masked Dword Write MSD			
		Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If Per Slot Offset is set, the global offset is added to those offsets to form the overall offset.	
	Value	Name	
	[0-2047]		
	3:0	URB Opcode	
	Format:	Opcode	
	Value	Name	Description
	7	URB_SIMD8_WRITE [Default]	SIMD8 URB Dword Write message. Writes 1..8 Dwords, based on RLEN and Channel Mask.

VD_CONTROL_STATE

VD_CONTROL_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This command can be used in HCPpipe. For HCP, it is selected with the Media Instruction Opcode "7h". Each command has assigned a media instruction command as defined in DWord 0, BitField 22:16. It will be different between HCP.</p> <p>This command is used to modify the control of HCP pipe. It can be inserted anywhere within a frame. It can be inserted multiple times within a frame as well.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
	26:23	Media Instruction Opcode	
		Default Value:	7h Codec/Engine Name for HCP
Format:		OpCode	
Codec/Engine Name = HCP = 7h			
22:16	Media Instruction Command		
	Default Value:	Ah VD_CONTROL_STATE	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
2h			
1..2	63:0	VD Control State Body	
		Format:	VD_CONTROL_STATE_BODY

VD_PIPELINE_FLUSH

VD_PIPELINE_FLUSH		
Source:	VideoCS	
Length Bias:	2	
DWord	Bit	Description
0	31:29	Command Type
		Default Value: 3h PARALLEL_VIDEO_PIPE
		Format: OpCode
	28:27	Pipeline
		Default Value: 2h Media
		Format: OpCode
	26:23	Media Command Opcode
		Default Value: Fh Extended command
Format: OpCode		
22:21	SubOpcodeA	
	Default Value: 0h	
	Format: OpCode	
20:16	SubOpcodeB	
	Default Value: 0h	
	Format: OpCode	
15:12	Reserved	
	Format: MBZ	
11:0	DWORD_COUNT_n	
	Default Value: 0h Excludes DWord (0)	
	Format: =n	
	Total Length - 2	
1	31:21	Reserved
		Format: MBZ
	20	AVP pipeline command flush
	Format: U1	
	19	MFX pipeline command flush
Format: U1		
18	Reserved	
17	VD-ENC pipeline command flush	
	Format: U1	

VD_PIPELINE_FLUSH		
	16	HEVC pipeline command flush Format: U1
	15:6	Reserved Format: MBZ
	5	AVP pipeline Done Format: U1
	4	VD command/message parser Done Format: U1
	3	MFX pipeline Done Format: U1
	2	Reserved
	1	VD-ENC pipeline Done Format: U1
	0	HEVC pipeline Done Format: U1



VDENC_CONTROL_STATE

VDENC_CONTROL_STATE			
Source:		BSpec	
Length Bias:		2	
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline Type	
		Default Value:	2h
		Format:	OpCode
	26:23	Media Instruction Opcode	
Default Value:		1h Codec/Engine Name for VDENC	
Format:		OpCode	
Codec/Engine Name = VDNEC = 1h;			
22:16	Media Instruction Command		
	Default Value:	Bh VD_CONTROL_STATE for VDNEC	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	Dword Length		
	Format:	=n	
	(Excludes Dwords 0, 1).		
	Value	Name	
0h			
1	31:2	Reserved	
		Format:	MBZ
	1	VDenc Initialization This bit, when set, clears internal states for VDenc Pipe. This bit should be set once at the beginning of aframe.	
0	Reserved		
Format:	MBZ		

VDENC_DS_REF_SURFACE_STATE

VDENC_DS_REF_SURFACE_STATE			
Source:	VideoCS		
Length Bias:	2		
This command specifies the surface state parameters for the downscaled reference surfaces.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_COMMON
	26:23	Opcode	
		Default Value:	1h VDENC_PIPE
	22:21	SubOpA	
Default Value:		0h	
20:16	SubOpB		
	Default Value:	3h VDENC_DS_REF_SURFACE_STATE	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Format:	=n	
	Total Length - 2		
	Value	Name	Description
4h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)	
1	31:0	Reserved	
		Format:	MBZ
2..5	127:0	Dwords 2..5	
		Format:	VDENC_Surface_State_Fields
Four DWords that define the bit fields used by the three surface state commands.			

VDENC_PIPE_BUF_ADDR_STATE

VDENC_PIPE_BUF_ADDR_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This state command provides the memory base addresses for all row stores, Streamin/StreamOut, DMV buffer along with the uncompressed source, reference pictures and downscaled reference pictures required by the VDENC pipeline. All reference pixel surfaces in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. Same holds true for the down-scaled reference pictures too. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculate the corresponding memory location within the frame buffer directly.</p> <p>VDenc supports 3 Downscaled reference frames (2 fwd, 1 bwd) and 4 normal reference frames (3 fwd, 1 bwd). The driver will sort out the base address from the DPB table and populate the base addresses that map to the corresponding reference index for both DS references and normal reference frames.</p> <p>Each of the individual DS ref/ Normal ref frames have their own MOCS DW that corresponds to the respective base address. The only thing that is different in the MOCS DW amongst the DS reference frames is the MMCD controls (specified in bits [10:9] of the MOCS DW). Driver needs to ensure that the other bits need to be the same across the different DS ref frames. The same is applicable for the normal reference frames.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_COMMON
		Format:	OpCode
	26:23	Opcode	
		Default Value:	1h VDENC_PIPE
		Format:	OpCode
	22:21	SubOpA	
		Default Value:	0h
		Format:	OpCode
	20:16	SubOpB	
		Default Value:	4h VDENC_PIPE_BUF_ADDR_STATE
Format:		OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Format:	=n	
	Total Length - 2		

VDENC_PIPE_BUF_ADDR_STATE								
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>3Bh</td> <td>DWORD_COUNT_n [Default]</td> <td>Excludes DWord (0,1)</td> </tr> </tbody> </table>	Value	Name	Description	3Bh	DWORD_COUNT_n [Default]	Excludes DWord (0,1)
Value	Name	Description						
3Bh	DWORD_COUNT_n [Default]	Excludes DWord (0,1)						
1..3	95:0	<p>DS FWD REF0</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Down_Scaled_Reference_Picture</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>Memory Compression Enable bit should be set to zero .</p>	Format:	VDENC_Down_Scaled_Reference_Picture				
Format:	VDENC_Down_Scaled_Reference_Picture							
4..6	95:0	<p>DS FWD REF1</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Down_Scaled_Reference_Picture</td> </tr> </table> <p style="text-align: center;">Description</p> <p>This field is used for pointing to DSFWD reference picture for L0 Refidx 1 used for Stage 1 of Motion Estimation.</p> <p>AVC Mode: This field must be programmed to have the same base address as DS FWD REF0 [List 0, refidx 0] when VDEncPerfMode is enabled in the VDENC_IMAGE_STATE. In Normal Mode, This field points to DS FWD REF1 [List 0, refidx 1]</p> <p>In HEVC / VP9 Mode: This field points to Stage 1 (8X Downscaled) base address[List 0, refidx 1] for LDB frame type in HEVC / P - frame in VP9.</p> <p>HEVC Mode: This field points to Stage1 (8X DS BWD REF 0) [List 1, refidx 0] for RA frame type in HEVC.</p> <p style="text-align: center;">Programming Notes</p> <p>Memory Compression Enable bit should be set to zero.</p>	Format:	VDENC_Down_Scaled_Reference_Picture				
Format:	VDENC_Down_Scaled_Reference_Picture							
7..9	95:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
10..12	95:0	<p>Original Uncompressed Picture</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Original_Uncompressed_Picture</td> </tr> </table>	Format:	VDENC_Original_Uncompressed_Picture				
Format:	VDENC_Original_Uncompressed_Picture							
13..15	95:0	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ							
16..18	95:0	<p>Row Store Scratch Buffer</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Row_Store_Scratch_Buffer_Picture</td> </tr> </table>	Format:	VDENC_Row_Store_Scratch_Buffer_Picture				
Format:	VDENC_Row_Store_Scratch_Buffer_Picture							
19..21	95:0	<p>Colocated MV Read Buffer</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Colocated_MV_Picture</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>This is ignored by HW since B-Frames are not supported.</p>	Format:	VDENC_Colocated_MV_Picture				
Format:	VDENC_Colocated_MV_Picture							

		VDENC_PIPE_BUF_ADDR_STATE	
22..24	95:0	FWD REF0	
		Format:	VDENC_Reference_Picture
		Programming Notes	
		<p>The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture 7 is used as reference for current frame. VDENC FWD REF0 should be programmed with [7] HCP_REF_IDX (L0) should map to reference picture 7 for reference index 0 in Reference picture list 0.</p> <p>This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals 0 and TBC is enabled.</p>	
25..27	95:0	FWD REF1	
		Format:	VDENC_Reference_Picture
		Programming Notes	
		<p>The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture6 is used as reference for current frame. VDENC FWD REF1 should be programmed with [6] HCP_REF_IDX (L0) should map to reference picture6 for reference index1 in Reference picture list 0.</p> <p>This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals 1 and TBC is enabled.</p>	
28..30	95:0	FWD REF2	
		Format:	VDENC_Reference_Picture
		Programming Notes	
		<p>The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture0 is used as reference for current frame. VDENC FWD REF1 should be programmed with [0] HCP_REF_IDX (L0) should map to reference picture0 for reference index2 in Reference picture list 0.</p> <p>This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals2 and TBC is enabled.</p>	
31..33	95:0	BWD REF0	
		Format:	VDENC_Reference_Picture
		Programming Notes	

VDENC_PIPE_BUF_ADDR_STATE						
		<p>The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture1 is used as only backward reference. VDENC BWD REF0 should be programmed with [1] HCP_REF_IDX (L1) should map to reference picture1 for reference index0 in Reference picture list 1.</p> <p>This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals3 and TBC is enabled.</p>				
34..36	95:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
37..39	95:0	<p>DS FWD REF0 4X</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>VDENC_Down_Scaled_Reference_Picture</td> </tr> </table> <p>For HEVC / VP9 VDEnc, this field indicates the 4X DS surface base address used in the stage2 ME for Reference frame0.</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </table> <p>Memory Compression Enable bit should be set to zero</p>	Format:	VDENC_Down_Scaled_Reference_Picture	Programming Notes	
Format:	VDENC_Down_Scaled_Reference_Picture					
Programming Notes						
40..42	95:0	<p>DS FWD REF1 4X</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>VDENC_Down_Scaled_Reference_Picture</td> </tr> </table> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Description</th> </tr> </table> <p>HEVC Mode: In RApictures, this field pointstoDS BWD REF0 (List 1 refidx 0)used in the stage2 HME. HEVC Mode / VP9 Mode: In LDB picture of HEVC and VP9 Mode, this field points to DS FWD REF1, Down scaled Forward Reference 1(List 0 refidx 1), used in stage 2 HME</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> </table> <p>Memory Compression Enable bit should be set to zero</p>	Format:	VDENC_Down_Scaled_Reference_Picture	Description	Programming Notes
Format:	VDENC_Down_Scaled_Reference_Picture					
Description						
Programming Notes						
43..45	95:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					
62..64	95:0	<p>VDENC Tile Row store Buffer</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>VDENC_Row_Store_Scratch_Buffer_Picture</td> </tr> </table> <p>Specifies 64 byte aligned Tile Row store buffer. This surface is a linear surface. The total size of the surface to be allocated is $32 * (W + 31) / 32 * 4$ Bytes, Where W is width of the picture in pixels. Each 32x32 needs one CL for row store. To support tile replay we need tile write rowstore and tile read rowstore.</p>	Format:	VDENC_Row_Store_Scratch_Buffer_Picture		
Format:	VDENC_Row_Store_Scratch_Buffer_Picture					
65..67	95:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					



VDENC_PIPE_BUF_ADDR_STATE		
68..70	95:0	Reserved
		Format: MBZ
86..88	95:0	Reserved
		Format: MBZ

VDENC_PIPE_MODE_SELECT

VDENC_PIPE_MODE_SELECT				
Source:	VideoCS			
Length Bias:	2			
<p>Specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis.</p> <p>The VDENC_PIPE_MODE_SELECT command specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis. It also configures the hardware pipeline according to the active encoder/decoder operating mode for encoding/decoding the current picture.</p>				
DWord	Bit	Description		
0	31:29	Command Type		
		Default Value: 3h PARALLEL_VIDEO_PIPE		
	Format: OpCode			
	28:27	Pipeline		
		Default Value: 2h MFX_COMMON		
	Format: OpCode			
	26:23	Opcode		
		Default Value: 1h VDENC_PIPE		
Format: OpCode				
22:21	SubOpA			
	Default Value: 0h			
Format: OpCode				
20:16	SubOpB			
	Default Value: 0h VDENC_PIPE_MODE_SELECT			
Format: OpCode				
15:12	Reserved			
Format: MBZ				
11:0	DWord Length			
	Format: =n			
	Total Length - 2			
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>4</td> <td></td> </tr> </tbody> </table>	Value	Name	4
Value	Name			
4				
1	31	Reserved		
	Format: MBZ			
30:28	Reserved			
Format: MBZ				

VDENC_PIPE_MODE_SELECT		
27:26	Reserved	
25	Reserved	
24:23	Reserved	
22:21	Reserved	
20	Reserved	
19	Reserved	
	Format:	MBZ
18	isRandomAccess bit This bit needs to be set to 1 in B-frames.	
17	Reserved	
16:15	PAK chroma sub-sampling type	
	Format:	U2
	This field is applicable only in HEVC and VP9. In AVC, this field is ignored.	
	Value	Name Description
	0	Reserved
	1	4:2:0 [Default] Used for Main8 and Main10 HEVC, VP9 profile0.
	3	4:4:4 HEVC RExt 444, VP9 444 profiles.
14:10	Reserved	
	Format:	MBZ
9	Reserved	
8	Reserved	
7	Reserved	
6	Reserved	
	Format:	MBZ
5	Reserved	
4	Scalability Mode	
	Format:	Enable
	When this is set, frame is encoded using multiple VDENC pipes. This bit is not used by HW.	
3:0	Standard Select	
	Format:	U4

VDENC_PIPE_MODE_SELECT			
		Value	Name
		2	AVC
		[4-15]	Reserved
2	31:28	Reserved	
		Format:	MBZ
	27:24	Reserved	
	23:20	Reserved	
	19:16	Reserved	
	15:12	Reserved	
		Format:	MBZ
	11:8	Reserved	
	7:4	Reserved	
	3	Reserved	
2:1	Reserved		
0	Reserved		
3	31:28	Reserved	
		Format:	MBZ
	27:24	Reserved	
	23:20	Reserved	
	19:16	Reserved	
	15:12	Reserved	
		Format:	MBZ
	11:8	Reserved	
	7:4	Reserved	
	3:2	Reserved	
Format:	MBZ		
	Reserved		
1	Reserved		
0	Reserved		
4	31:0	Reserved	
		Format:	MBZ
5	31:0	Reserved	
		Format:	MBZ

VDENC_REF_SURFACE_STATE

VDENC_REF_SURFACE_STATE			
Source:	VideoCS		
Length Bias:	2		
This command specifies the surface state parameters for the normal reference surfaces.			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_COMMON
	26:23	Opcode	
		Default Value:	1h VDENC_PIPE
	22:21	SubOpA	
Default Value:		0h	
20:16	SubOpB		
	Default Value:	2h VDENC_REF_SURFACE_STATE	
15:12	Reserved		
	Format:	MBZ	
11:0	DWord Length		
	Format:	=n	
	Total Length - 2		
	Value	Name	Description
4h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)	
1	31:0	Reserved	
		Format:	MBZ
2..5	127:0	Dwords 2..5	
		Format:	VDENC_Surface_State_Fields
		Four DWords that define the bit fields used by the three surface state commands.	
		Programming Notes	
The height and width fields in surface state commands are not used by HW.			

VDENC_SRC_SURFACE_STATE

VDENC_SRC_SURFACE_STATE			
Source:	VideoCS		
Length Bias:	2		
<p>This command specifies the uncompressed original input picture to be encoded. The actual base address is defined in the VDENC_PIPE_BUF_ADDR_STATE. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. T</p> <p>For non pixel data, such as row stores, DMV and streamin/out, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This mechanism is chosen over the pixel surface type because of their variable record sizes.</p> <p>All row store surfaces are linear surface. Their addresses are programmed in VDenc_Pipe_Buf_Base_State.</p>			
DWord	Bit	Description	
0	31:29	Command Type	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_COMMON
		Format:	OpCode
	26:23	Opcode	
		Default Value:	1h VDENC_PIPE
		Format:	OpCode
	22:21	SubOpA	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpB		
	Default Value:	1h VDENC_SRC_SURFACE_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	11:0	DWord Length	
		Format:	=n
	Total Length - 2		
	Value	Name	Description
4h	DWORD_COUNT_n [Default]	Excludes DWord (0,1)	
1	31:0	Reserved	

VDENC_SRC_SURFACE_STATE							
	<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
2..5	<table border="1"> <tr> <td>127:0</td> <td> <p>Dwords 2..5</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Surface_State_Fields</td> </tr> </table> <p>Four DWords that define the bit fields used by the three surface state commands.</p> <table border="1"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> </table> <p>The Width field programmed in DW 2 bit 17:4 should match frame width in pixel, ignoring the comment in Width field</p> </td> </tr> </table>	127:0	<p>Dwords 2..5</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Surface_State_Fields</td> </tr> </table> <p>Four DWords that define the bit fields used by the three surface state commands.</p> <table border="1"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> </table> <p>The Width field programmed in DW 2 bit 17:4 should match frame width in pixel, ignoring the comment in Width field</p>	Format:	VDENC_Surface_State_Fields	Programming Notes	
127:0	<p>Dwords 2..5</p> <table border="1"> <tr> <td>Format:</td> <td>VDENC_Surface_State_Fields</td> </tr> </table> <p>Four DWords that define the bit fields used by the three surface state commands.</p> <table border="1"> <tr> <td colspan="2" style="text-align: center;">Programming Notes</td> </tr> </table> <p>The Width field programmed in DW 2 bit 17:4 should match frame width in pixel, ignoring the comment in Width field</p>	Format:	VDENC_Surface_State_Fields	Programming Notes			
Format:	VDENC_Surface_State_Fields						
Programming Notes							

VDENC_WEIGHTSOFFSETS_STATE

VDENC_WEIGHTSOFFSETS_STATE			
Source:	BSpec		
Length Bias:	2		
DWord	Bit	Description	
0	31:29	Command_Type	
		Default Value:	03h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	Pipeline	
		Default Value:	2h MFX_COMMON
		Format:	OpCode
	26:23	OpCode	
		Default Value:	1h VDENC_PIPE
		Format:	OpCode
	22:21	SubOpA	
Default Value:		0h	
Format:		OpCode	
20:16	SubOpB		
	Default Value:	8h VDENC_AVC_WEIGHTSOFFSET_STATE	
	Format:	OpCode	
15:12	Reserved		
	Format:	MBZ	
11:0	DW Length		
	Format:	=n	
		Total Length - 2	
1	31:24	Offset Forward Reference1	
		Format:	S7
	<p>The weights and offset fields defined in this DW are used for both HEVC and AVC.</p> <p>This parameter indicates the explicit weighting factor applied to the luma prediction value of forward reference1 (refidx=1) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>		
23:16	Weights Forward Reference1		
	Format:	S7	
<p>This parameter indicates the explicit weighting factor applied to the luma prediction value of forward reference1 (refidx=1) frame.</p>			

VDENC_WEIGHTSOFFSETS_STATE				
		This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.		
	15:8	<p>Offset Forward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the luma prediction value of forward reference0 (refidx=0) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7			
	7:0	<p>Weights Forward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the luma prediction value of forward reference0 (refidx=0) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7			
<p>2</p> <p>The weights and offset fields defined in this DW are used for both HEVC and AVC.</p>	31:24	<p>Offset Backward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the luma prediction value of forward reference0 (refidx=0) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127. This is also used by stage2 HME for the 4X DS search in HEVC</p>	Format:	S7
	Format:	S7		
	23:16	<p>Weights Backward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the luma prediction value of forward reference0 (refidx=0) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -127 to +127. The derived Delta Value (Weights Backward Reference0 (1«Log2WeightDenomLuma)) also needs to be in the range from -128 to +127. This is also used by stage2 HME for the 4X DS search in HEVC</p>	Format:	S7
Format:	S7			
15:8	<p>Offset Forward Reference2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the luma</p>	Format:	S7	
Format:	S7			

VDENC_WEIGHTSOFFSETS_STATE			
	<p>prediction value of forward reference2 (refidx=2) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>		
	<p>7:0 Weights Forward Reference2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the luma prediction value of forward reference2 (refidx=2) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7		
<p>3 The weights and offset fields defined in this DW are used for both HEVC and AVC.</p>	<p>31:24 CB Chroma Offset Forward Reference1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference1 (refidx=1) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
	Format:	S7	
	<p>23:16 CBChromaWeights Forward Reference1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference1 (refidx=1) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
	Format:	S7	
<p>15:8 CB Chroma Offset Forward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7	
Format:	S7		
<p>7:0 CB Chroma Weights Forward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS).</p>	Format:	S7	
Format:	S7		

VDENC_WEIGHTSOFFSETS_STATE				
		Range is from -128 to +127.		
<p style="text-align: center;">4</p> <p>The weights and offset fields defined in this DW are used for both HEVC and AVC.</p>	31:24	<p>CB Chroma Offset Backward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127. This is also used by stage2 HME for the 4X DS search in HEVC</p>	Format:	S7
	Format:	S7		
	23:16	<p>Chroma Weights Backward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -127 to +127. The derived Delta Value (Weights Backward Reference0 (1«Log2WeightDenomLuma)) also needs to be in the range from -128 to +127. This is also used by stage2 HME for the 4X DS search in HEVC</p>	Format:	S7
	Format:	S7		
15:8	<p>CB Chroma Offset Forward Reference2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference2 (refidx=2) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7	
Format:	S7			
7:0	<p>CB Chroma Weights Forward Reference2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CB Chroma prediction value of forward reference2 (refidx=2) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7	
Format:	S7			
<p style="text-align: center;">5</p> <p>The weights and offset fields defined in this DW are used for both HEVC and AVC.</p>	31:24	<p>CR Chroma Offset Forward Reference1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 80%;">Format:</td> <td style="width: 20%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CR Chroma prediction value of forward reference1 (refidx=1) frame. This should be programmed to zero if explicit weighting is disabled for the</p>	Format:	S7
	Format:	S7		

VDENC_WEIGHTSOFFSETS_STATE			
	<p>current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>		
	<p>23:16 CR ChromaWeights Forward Reference1</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CR Chroma prediction value of forward reference1 (refidx=1) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7		
	<p>15:8 CR Chroma Offset Forward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CR Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7		
	<p>7:0 CR Chroma Weights Forward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CR Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7		
6	<p>31:24 CR Chroma Offset Backward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CR Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127. This is also used by stage2 HME for the 4X DS search in HEVC</p>	Format:	S7
Format:	S7		
6	<p>23:16 CR Chroma Weights Backward Reference0</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CR Chroma prediction value of forward reference0 (refidx=0) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -127 to +127. The derived Delta Value (Weights Backward</p>	Format:	S7
Format:	S7		

VDENC_WEIGHTSOFFSETS_STATE			
	<p>Reference0 ($1 \ll \text{Log2WeightDenomLuma}$) also needs to be in the range from -128 to +127. This is also used by stage2 HME for the 4X DS search in HEVC</p>		
15:8	<p>CR Chroma Offset Forward Reference2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the Cr Chroma prediction value of forward reference2 (refidx=2) frame. This should be programmed to zero if explicit weighting is disabled for the current super-slice (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7		
7:0	<p>CR Chroma Weights Forward Reference2</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S7</td> </tr> </table> <p>This parameter indicates the explicit weighting factor applied to the CR Chroma prediction value of forward reference2 (refidx=2) frame. This should be programmed to be equal to "1" if explicit weighting is disabled (indicated by the WeightedPredFlag in the PPS). Range is from -128 to +127.</p>	Format:	S7
Format:	S7		

VEBOX_STATE

VEBOX_STATE						
Source:	VideoEnhancementCS					
Length Bias:	2					
<p>This command controls the internal functions of the VEBOX. This command has a set of indirect state buffers:</p> <ul style="list-style-type: none"> • DN/DI state • IECP general state • IECP Gamut Expansion/Compression state • IECP Gamut Vertex Table state • Capture Pipe state 						
Adds the LACE LUT Table as an indirect state buffer.						
DWord	Bit	Description				
0	31:29	Command Type				
		Default Value: 3h PARALLEL_VIDEO_PIPE				
	Format: OpCode					
	28:27	Pipeline				
		Default Value: 2h Media				
	Format: OpCode					
	26:24	Command OpCode				
		Default Value: 4h VEBOX				
Format: OpCode						
23:21	SubOpcode A					
	Default Value: 0h					
Format: OpCode						
20:16	SubOpcode B					
	Default Value: 2h					
Format: OpCode						
15:12	Reserved					
	Format: MBZ					
11:0	DWord Length					
	Format: =n					
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>11h</td> <td></td> <td>(Excludes DWords 0, 1)</td> </tr> </tbody> </table>	Value	Name	Description	11h	
Value	Name	Description				
11h		(Excludes DWords 0, 1)				
1	31:25	State Surface Control Bits				

VEBOX_STATE							
	<p>All Indirect state buffers use state surface control bits, only exception being 3D LUT state buffer for which the state surface control bits are tied to 0. See definition under "VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS" Bits[6:0] is only used.</p>						
24	FP16 mode enable						
23	Reserved						
22	<p>Gamut Expansion Position If Gamut Expansion is enabled, it can be configured either in front or backend of the IECP pipe using this bit.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Gamut Expansion at the Backend of IECP pipe</td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Gamut Expansion at the Front of IECP pipe</td> </tr> </tbody> </table>	Value	Name	0b	Gamut Expansion at the Backend of IECP pipe	1b	Gamut Expansion at the Front of IECP pipe
Value	Name						
0b	Gamut Expansion at the Backend of IECP pipe						
1b	Gamut Expansion at the Front of IECP pipe						
21	<p>Forward Gamma Correction Enable</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>Enable</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>Single Pipe IECP Enable must also be set if this is enabled.</p> <p>When enabled the forward gamma will always be in front of the IECP pipe. In case disabled it will be always configured as Gamut expansion. Gamut Expansion, HDR and Forward Gamma Correction are mutually exclusive.</p>	Format:	Enable				
Format:	Enable						
20	<p>Scalar Mode When Scalar Mode is enabled, all other VEBOX functions must be disabled (DN/DI/DM/IECP/Chroma upsampling).</p>						
19	<p>Single Pipe Enable</p> <p>Indicates that the Capture Pipe features that only exist in a single pipe can be enabled.</p> <p>This bit must be set if any of the following features are enabled: Demosaic Denoise with one of the RGBA input formats IECP only mode with Forward Gamma Correction enabled with RGB input formats (All other modes are not supported in single pipe)</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Enable</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Default [Default]</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Note that the pixel throughput is 1/2 when this mode is selected. The Global IECP Enable must also be set.</p>	Value	Name	1	Enable	0	Default [Default]
Value	Name						
1	Enable						
0	Default [Default]						
18	<p>Disable Temporal Denoise Filter If set this bit will force the denoise filter to only use the spatial filter. This will eliminate the read of the previous denoise surface and STMM/Denoise History surface and the write of the current denoised surface and STMM/Denoise History surface.</p>						

VEBOX_STATE							
	<table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2"> <p>The Global IECP Enable or Demosaic Enable must be set along with this bit. This bit must be set if the input to Denoise is RGB. This bit must not be set if the Deinterlacer is enabled. This bit must be clear if both DN Enable=0 and Hot Pixel Filtering Enable=0. This bit must be set if Hot Pixel Filtering Enable=1 and both DN and DI are disabled.</p> </td> </tr> </table>	Programming Notes		<p>The Global IECP Enable or Demosaic Enable must be set along with this bit. This bit must be set if the input to Denoise is RGB. This bit must not be set if the Deinterlacer is enabled. This bit must be clear if both DN Enable=0 and Hot Pixel Filtering Enable=0. This bit must be set if Hot Pixel Filtering Enable=1 and both DN and DI are disabled.</p>			
Programming Notes							
<p>The Global IECP Enable or Demosaic Enable must be set along with this bit. This bit must be set if the input to Denoise is RGB. This bit must not be set if the Deinterlacer is enabled. This bit must be clear if both DN Enable=0 and Hot Pixel Filtering Enable=0. This bit must be set if Hot Pixel Filtering Enable=1 and both DN and DI are disabled.</p>							
17	<p>Disable Encoder Statistics If set this bit will disable writing the per block Encoder statistics. The memory format is not changed, so the area set aside for these statistics will still be there.</p>						
16	<p>LACE Correction Enable This bit enables the correction of the image according to the local ACE LUT tables. This is independent from the enable for the collection of LACE histograms.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2"> <p>LACE correction is only enabled if both this bit and the Global IECP Enable are set. The ACE Enable bit should also be set if this bit is set, since ACE correction can be used for part of the luma range instead of LACE.</p> </td> </tr> </table>	Programming Notes		<p>LACE correction is only enabled if both this bit and the Global IECP Enable are set. The ACE Enable bit should also be set if this bit is set, since ACE correction can be used for part of the luma range instead of LACE.</p>			
Programming Notes							
<p>LACE correction is only enabled if both this bit and the Global IECP Enable are set. The ACE Enable bit should also be set if this bit is set, since ACE correction can be used for part of the luma range instead of LACE.</p>							
15:14	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ				
Format:	MBZ						
13	<p>Hot Pixel Filtering Enable Enables hot pixel detection/filtering.</p>						
12	<p>Alpha Plane Enable Enables the reading of an independent Alpha plane. Mutually exclusive with Vignette Enable. If Alpha from State Select is set it overrides this bit.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2"> <p>IECP must also be enabled and output format must have alpha if this bit is enabled. Should be 0 if Alpha from State Select is 1.</p> </td> </tr> </table>	Programming Notes		<p>IECP must also be enabled and output format must have alpha if this bit is enabled. Should be 0 if Alpha from State Select is 1.</p>			
Programming Notes							
<p>IECP must also be enabled and output format must have alpha if this bit is enabled. Should be 0 if Alpha from State Select is 1.</p>							
11	<p>Vignette Enable Enables Vignette Correction surface read and correction in IECP. Mutually exclusive with Alpha Plane Enable.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2"> <p>Demosaic must also be enabled if this bit is enabled.</p> </td> </tr> </table>	Programming Notes		<p>Demosaic must also be enabled if this bit is enabled.</p>			
Programming Notes							
<p>Demosaic must also be enabled if this bit is enabled.</p>							
10	<p>Demosaic Enable The Demosaic will be used, and White balance statistics will be gathered. The Capture Pipe State Table will be read. This bit is mutually exclusive with DI Enable.</p> <table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2"> <p>IECP must also be enabled if this bit is enabled.</p> </td> </tr> </table>	Programming Notes		<p>IECP must also be enabled if this bit is enabled.</p>			
Programming Notes							
<p>IECP must also be enabled if this bit is enabled.</p>							
9:8	<p>DI Output Frames Indicates which frames to output in DI mode.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td>Output Both Frames</td> </tr> <tr> <td style="text-align: center;">01b</td> <td>Output Previous Frame Only</td> </tr> </tbody> </table>	Value	Name	00b	Output Both Frames	01b	Output Previous Frame Only
Value	Name						
00b	Output Both Frames						
01b	Output Previous Frame Only						

VEBOX_STATE									
	<table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">10b</td> <td>Output Current Frame Only</td> </tr> </table> <p style="text-align: center;">Programming Notes</p> <p>Field is ignored if DI Enable = 0. If Previous Frame Only or Current Frame Only are selected, then the LACE Single Histogram Set must not try to collect a histogram from the disabled frame.</p> <p>Field must be programmed to 10 (Output Current Frame Only) for DI First Frame.</p>	10b	Output Current Frame Only						
10b	Output Current Frame Only								
7:6	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
5	<p>DN/DI First Frame</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Indicates that this is the first frame of the stream, so previous clean is not available.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Not first field; previous clean surface state is valid</td> </tr> <tr> <td style="text-align: center;">1</td> <td>First field; previous clean surface state is invalid</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>If both DN and DI are disabled, this bit must be 0.</p>	Format:	Enable	Value	Name	0	Not first field; previous clean surface state is valid	1	First field; previous clean surface state is invalid
Format:	Enable								
Value	Name								
0	Not first field; previous clean surface state is valid								
1	First field; previous clean surface state is invalid								
4	<p>DI Enable</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Deinterlacer is bypassed if this is disabled: the output is the same as the input (same as a 2:2 cadence). FMD and STMM are not calculated and the values in the response message are 0.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Do not calculate DI</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Calculate DI</td> </tr> </tbody> </table>	Format:	Enable	Value	Name	0	Do not calculate DI	1	Calculate DI
Format:	Enable								
Value	Name								
0	Do not calculate DI								
1	Calculate DI								
3	<p>DN Enable</p> <table border="1" style="width: 100%;"> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>Denoise is bypassed if this is low - BNE is still calculated and output, but the denoised fields are not. VDI does not read in the denoised previous frame but uses the pointer for the original previous frame.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Do not denoise frame</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Denoise frame</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>If DN and/or Hotpixel are the only functions enabled then the only output is the Denoised Output which is the same surface format as the input. To get a format conversion with DN only, enable the Global IECP bit, but disable all the individual functions. The IECP output uses the output surface format.</p> <p>If DN is used with RGB then the Global IECP Enable must also be</p>	Format:	Enable	Value	Name	0	Do not denoise frame	1	Denoise frame
Format:	Enable								
Value	Name								
0	Do not denoise frame								
1	Denoise frame								

VEBOX_STATE		
2	2	Global IECP Enable Indicates if any of the IECP features is enabled. If this is disabled then no state will be read from any of the state pointers. If set then the IECP state will be read.
	1	Color Gamut Compression Enable Indicates if the Gamut Compression feature is enabled. If set then the Gamut State will be read. VEB_VERTEXABLE_STATE is only needed if this bit is set.
	0	Color Gamut Expansion Enable Indicates if the Gamut Expansion feature is enabled. If set then the Gamut State will be read. This can be enabled only if Single pipe enable is disabled.
2	31:12	DN/DI State Pointer Low Format: GraphicsAddress[31:12] Bits 31:12 of the starting address of the DN/DI State buffer. This points to a buffer containing the 10 Dwords of the DN/DI state. When Scalar mode is enabled this pointer is used for Scalar state table.
	11:0	Reserved Format: MBZ
3	31:16	Reserved Format: MBZ
	15:0	DN/DI State Pointer High Format: GraphicsAddress[47:32] Bits 47:32 of the starting address of the DN/DI State Buffer. When Scalar mode is enabled this pointer is used for Scalar state table.
4	31:12	IECP State Pointer Low Format: GraphicsAddress[31:12] Bits 31:12 of the starting address of the IECP State buffer. This points to a buffer containing the 64 Dwords of IECP state.
	11:0	Reserved Format: MBZ
5	31:16	Reserved Format: MBZ
	15:0	IECP State Pointer High Format: GraphicsAddress[47:32] Bits 47:32 of the starting address of the IECP State Buffer Table.
6	31:12	Gamut/HDR State Pointer Low Format: GraphicsAddress[31:12]

VEBOX_STATE				
		<p>Bits 31:12 of the starting address of the State Buffer.</p> <p>If Gamut Expansion is enabled, this points to a buffer containing the Gamut Expansion Gamma Correction state.</p> <p>If HDR is enabled, this points to a buffer containing the HDR state.</p>		
	11:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
Format:	MBZ			
7	31:16	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	<p>Gamut/HDR State Pointer High</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">GraphicsAddress[47:32]</td> </tr> </table> <p>Bits 47:32 of the starting address of the Gamut/HDR State Buffer.</p>	Format:	GraphicsAddress[47:32]	
Format:	GraphicsAddress[47:32]			
8	31:12	<p>Vertex Table State Pointer Low</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">GraphicsAddress[31:12]</td> </tr> </table> <p>Bits 31:12 of the starting address of the Vertex Table. This points to a buffer containing the 512 Dwords of the Gamut Compression Vertex Table.</p>	Format:	GraphicsAddress[31:12]
	Format:	GraphicsAddress[31:12]		
11:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
9	31:16	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	<p>Vertex Table State Pointer High</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">GraphicsAddress[47:32]</td> </tr> </table> <p>Bits 47:32 of the starting address of the Vertex State Buffer.</p>	Format:	GraphicsAddress[47:32]	
Format:	GraphicsAddress[47:32]			
10	31:12	<p>Capture Pipe State Pointer Low</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">GraphicsAddress[31:12]</td> </tr> </table> <p>Bits 31:12 of the starting address of the Capture Pipe State Table. This points to a buffer containing the X Dwords of the Capture Pipe State.</p>	Format:	GraphicsAddress[31:12]
	Format:	GraphicsAddress[31:12]		
11:0	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ	
Format:	MBZ			
11	31:16	<p>Reserved</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td style="width: 40%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	<p>Capture Pipe State Pointer High</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">GraphicsAddress[47:32]</td> </tr> </table> <p>Bits 47:32 of the starting address of the Capture Pipe State Table.</p>	Format:	GraphicsAddress[47:32]	
Format:	GraphicsAddress[47:32]			

VEBOX_STATE												
12	31:12	LACE LUT Table State Pointer Low Format: GraphicsAddress[31:12] Bits [31:12] of the starting address of the LACE Look-up Tables.										
	11:0	Reserved Format: MBZ										
13	31:30	Arbitration Priority Control - For LACE LUT Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Highest priority</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Second highest priority</td> </tr> <tr> <td style="text-align: center;">2</td> <td>Third highest priority</td> </tr> <tr> <td style="text-align: center;">3</td> <td>Lowest priority</td> </tr> </tbody> </table>	Value	Name	0	Highest priority	1	Second highest priority	2	Third highest priority	3	Lowest priority
	Value	Name										
	0	Highest priority										
	1	Second highest priority										
	2	Third highest priority										
3	Lowest priority											
29:16	Reserved Format: MBZ											
15:0	LACE LUT Table State Pointer High Format: GraphicsAddress[47:32] Bits [47:32] of the starting address of the LACE Look-up Tables.											
14..15	63:12	Gamma Correction Values Address Format: VIRTUAL_ADDR[63:12] Specifies the 4K byte aligned address reading the Gamma Correction Values in case enabled.										
	11:0	Reserved Format: MBZ										
16	31:12	3D LUT State Pointer Low Format: GraphicsAddress[31:12] Bits [31:12] of the starting address of the 3D LUT.										
	11:0	Reserved Format: MBZ										
17	31:30	Arbitration Priority Control - For 3D LUT Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Highest Priority</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Second highest priority</td> </tr> </tbody> </table>	Value	Name	0	Highest Priority	1	Second highest priority				
	Value	Name										
	0	Highest Priority										
1	Second highest priority											

VEBOX_STATE											
	<table border="1"> <tr> <td style="width: 100px;">2</td> <td>Third highest priority</td> </tr> <tr> <td>3</td> <td>Lowest priority</td> </tr> </table>	2	Third highest priority	3	Lowest priority						
2	Third highest priority										
3	Lowest priority										
29	Reserved										
28:24	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
23:22	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
21:16	3D LUT MOCS table These are surface control bits for VEBOX 3DLUT data requests to GAV										
15:0	3D LUT State Pointer High <table border="1"> <tr> <td>Format:</td> <td>GraphicsAddress[47:32]</td> </tr> </table> Bits [47:32] of the starting address of the 3D LUT.	Format:	GraphicsAddress[47:32]								
Format:	GraphicsAddress[47:32]										
18	31 3D LUT Enable <table border="1"> <tr> <td>Default Value:</td> <td>0</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> 3D LUT is required only if this is enabled. <table border="1" style="background-color: #e6f2ff;"> <tr> <th colspan="2" style="text-align: center;">Restriction</th> </tr> <tr> <td colspan="2" style="text-align: center;">The frame height needs to be multiple of 8 when enabling 3dlut in VEBOX dual pipe mode.</td> </tr> </table>	Default Value:	0	Format:	Enable	Restriction		The frame height needs to be multiple of 8 when enabling 3dlut in VEBOX dual pipe mode.			
	Default Value:	0									
	Format:	Enable									
	Restriction										
	The frame height needs to be multiple of 8 when enabling 3dlut in VEBOX dual pipe mode.										
	30:29 3D LUT Size <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>33x33x33</td> </tr> <tr> <td>01b</td> <td>17x17x17</td> </tr> <tr> <td>10b</td> <td>65x65x65</td> </tr> </tbody> </table>	Format:	U2	Value	Name	00b	33x33x33	01b	17x17x17	10b	65x65x65
	Format:	U2									
	Value	Name									
	00b	33x33x33									
	01b	17x17x17									
10b	65x65x65										
28:23 Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ										
22:16 Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ										
15:14 Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ									
Format:	MBZ										
13:12 Frame statistics ID <table border="1"> <tr> <td>Format:</td> <td>U2</td> </tr> </table> This field specifies the Statistics Surface ID number to the VEBOX to writeout the frame statistics.	Format:	U2									
Format:	U2										
11	Bypass Chroma Downsampling										

VEBOX_STATE

	Format:	U1
	When enabled will drop chroma samples at odd position and not use the co-sited offsets.	
	Value	Name
	0	[Default]
10	Bypass Chroma Upsampling	
	Format:	U1
	When enabled will replicate chroma samples at odd position and not use the co-sited offsets.	
	Value	Name
	0	[Default]
9:7	Chroma Downsampling Co-Sited Vertical Offset	
	Format:	U3
	Value	Name
	0	[Default]
	[0,2]	Valid Range
6:5	Chroma Downsampling Co-Sited Horizontal Offset	
	Format:	U2
	Value	Name
	0	[Default]
	[0,2]	Valid Range
4:2	Chroma Upsampling Co-Sited Vertical Offset	
	Format:	U3
	Value	Name
	0	[Default]
	[0,4]	Valid Range
1:0	Chroma Upsampling Co-Sited Horizontal Offset	
	Format:	U2
	Value	Name
	0	[Default]
	[0,1]	Valid Range



VEBOX_SURFACE_STATE

VEBOX_SURFACE_STATE	
Source:	VideoEnhancementCS
Length Bias:	2
<p>The input and output data containers accessed are called "surfaces". Surface state is sent to VEBOX via an inline state command rather than using binding tables. SURFACE_STATE contains the parameters defining each surface to be accessed, including its size, format, and offsets to its sub surfaces. The surface's base address is in the execution command. Despite having multiple input and output surfaces, we limit the number of surface states to one for input surfaces and one for output surfaces. The other surfaces are derived from the input/output surface states.</p>	
<p>The Current Frame Input surface uses the Input SURFACE_STATE</p>	
<p>The Previous Denoised Input surface uses the Input SURFACE_STATE. (For 16-bit Bayer pattern inputs this will be 16-bit.)</p>	
<p>The Current Denoised Output surface uses the Input SURFACE_STATE. (For 16-bit Bayer pattern inputs this will be 16-bit.)</p>	
<p>The STMM/Noise History Input surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.</p>	
<p>The STMM/Noise History Output surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.</p>	
<p>The Current Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.</p>	
<p>The Previous Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.</p>	
<p>The FMD per block output / per Frame Output surface uses the Linear SURFACE_STATE (see note below).</p>	
<p>The Alpha surface uses the Linear A8 SURFACE_STATE with Width/Height equal to Input Surface. Pitch is width rounded to next 64.</p>	
<p>The Skin Score surface uses the Output SURFACE_STATE.</p>	
<p>The STMM height is the same as the Input Surface height except when the input Surface Format is Bayer Pattern and the Bayer Pattern Offset is 10 or 11, in which case the height is the input height + 4. For Bayer pattern inputs when the Bayer Pattern Offset is 10 or 11, the Current Denoised Output/Previous Denoised Input will also have a height which is the input height + 4. For Bayer pattern inputs only the Current Denoised Output/Previous Denoised Input are in Tile-Y.</p>	
<p>The linear surface for FMD statistics is linear (not tiled). The height of the per block statistics is $(\text{Input Height} + 3) / 4$ - the Input Surface height in pixels is rounded up to the next even 4 and divided by 4. The width of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 16 bytes. The pitch of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 64 bytes.</p>	
<p>The STMM surfaces must be identical to the Input surface except for the tiling mode must be Tile-Y and the pitch is specified in DW7. The pitch for the Current Denoised Output/Previous Denoised Input is specified in DW7. The width and height must be a multiple of 4 rounded up from the input height.</p>	
<p>The Vignette Correction surface uses the Linear 16-bit SURFACE_STATE with : $\text{Width} = (\text{Ceil}(\text{Image Width} / 4) + 1) * 4$</p>	

VEBOX_SURFACE_STATE

Height= Ceil(Image Height / 4) + 1
 Pitch in bytes is (vignette width *2) rounded to the next 64

Programming Notes

VEBOX may write to memory between the surface width and the surface pitch for output surfaces.

VEBOX can support a frame level X/Y offset which allows processing of 2 side-by-side frames for certain 3D video formats.

The X/Y Offset for Frame state applies only to the Current Frame Input and the Current Deinterlaced/IECP Frame Output and Previous Deinterlaced/IECP Frame Output. The statistics surfaces, the denoise feedback surfaces and the alpha/vignette surfaces have no X/Y offsets.

For 8bit Alpha input, when converted to 16bit output, the 8 bit alpha value is replicated to both the upper and lower 8 bits to form the 16 bit alpha value.

Skin Score Output Surface uses the same tiling format as the Output surface.

DWord	Bit	Description				
0	31:29	Command Type				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>3h PARALLEL_VIDEO_PIPE</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	3h PARALLEL_VIDEO_PIPE	Format:	OpCode
	Default Value:	3h PARALLEL_VIDEO_PIPE				
	Format:	OpCode				
	28:27	Media Command Pipeline				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>2h Media</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	2h Media	Format:	OpCode
	Default Value:	2h Media				
	Format:	OpCode				
	26:24	Media Command OpCode				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>4h VEBOX</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	4h VEBOX	Format:	OpCode
Default Value:	4h VEBOX					
Format:	OpCode					
23:21	SubOpCode A					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>0h VEBOX</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h VEBOX	Format:	OpCode	
Default Value:	0h VEBOX					
Format:	OpCode					
20:16	SubOpCode B					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>0h VEBOX</td> </tr> <tr> <td>Format:</td> <td>OpCode</td> </tr> </table>	Default Value:	0h VEBOX	Format:	OpCode	
Default Value:	0h VEBOX					
Format:	OpCode					
15:12	Reserved					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ			
Format:	MBZ					
11:0	DWord Length					
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>=n</td> </tr> </table>	Format:	=n			
	Format:	=n				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 45%;">Name</th> <th style="width: 40%;">Description</th> </tr> </thead> <tbody> <tr> <td>7h</td> <td>DWORD_COUNT_n [Default]</td> <td>(Excludes DWords 0, 1)</td> </tr> </tbody> </table>	Value	Name	Description	7h	DWORD_COUNT_n [Default]	(Excludes DWords 0, 1)
Value	Name	Description				
7h	DWORD_COUNT_n [Default]	(Excludes DWords 0, 1)				
1	31:1	Reserved				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ					

VEBOX_SURFACE_STATE																																	
0	<p>Surface Identification Specifies which set of surfaces this command refers to:</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Output surface (all except the Denoised Current output surface)</td> </tr> <tr> <td style="text-align: center;">0</td> <td>Input surface and Denoised Current Output Surface</td> </tr> </tbody> </table>	Value	Name	1	Output surface (all except the Denoised Current output surface)	0	Input surface and Denoised Current Output Surface																										
Value	Name																																
1	Output surface (all except the Denoised Current output surface)																																
0	Input surface and Denoised Current Output Surface																																
2	<p>31:18 Height</p> <p>Format: U14</p> <p>This field specifies the height of the surface in units of pixels. For PLANAR surface formats, this field indicates the height of the Y (luma) plane.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> <th style="text-align: center;">Exists If</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[15, 16383]</td> <td></td> <td>representing heights [16,16384]</td> <td></td> </tr> <tr> <td style="text-align: center;">[15, 8191]</td> <td></td> <td></td> <td>//Scalar Enabled - For Input surface only</td> </tr> <tr> <td style="text-align: center;">[63, 2047]</td> <td></td> <td></td> <td>//Scalar + SFC Enabled - For Input surface only</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>Height (field value + 1) must be a multiple of 2 for PLANAR_420 surfaces. Height (field value + 1) must be a multiple of 2 when the deinterlace function is enabled (field mode) or when the denoise function is enabled with Progressive DN = 0. It must be a multiple of 4 when interleaved deinterlace/denoise and PLANAR_420 are both being used. VEBOX supports a minimum height of 16.</p> <p>Height (field value + 1) must be a multiple of 2 for Bayer surfaces.</p> <p>17:4 Width</p> <p>Format: U14</p> <p>This field specifies the width of the surface in units of pixels. For PLANAR surface formats, this field indicates the width of the Y (luma) plane.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> <th style="text-align: center;">Exists If</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">[63,16383]</td> <td></td> <td>representing widths [64,16384]</td> <td></td> </tr> <tr> <td style="text-align: center;">[63,8191]</td> <td></td> <td></td> <td>//Scalar Enabled - For Input surface only</td> </tr> <tr> <td style="text-align: center;">[63,2047]</td> <td></td> <td></td> <td>//Scalar and SFC Enabled - For Input Surface only</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). Width (field value + 1) must be a multiple of 2 for PLANAR_420, PLANAR_422, and all YCRCB_* surfaces, and must be a multiple of 4 for PLANAR_411 surfaces. VEBOX supports a minimum width of 64</p> <p>3:0 Reserved</p>	Value	Name	Description	Exists If	[15, 16383]		representing heights [16,16384]		[15, 8191]			//Scalar Enabled - For Input surface only	[63, 2047]			//Scalar + SFC Enabled - For Input surface only	Value	Name	Description	Exists If	[63,16383]		representing widths [64,16384]		[63,8191]			//Scalar Enabled - For Input surface only	[63,2047]			//Scalar and SFC Enabled - For Input Surface only
Value	Name	Description	Exists If																														
[15, 16383]		representing heights [16,16384]																															
[15, 8191]			//Scalar Enabled - For Input surface only																														
[63, 2047]			//Scalar + SFC Enabled - For Input surface only																														
Value	Name	Description	Exists If																														
[63,16383]		representing widths [64,16384]																															
[63,8191]			//Scalar Enabled - For Input surface only																														
[63,2047]			//Scalar and SFC Enabled - For Input Surface only																														

VEBOX_SURFACE_STATE																																																					
		Format: MBZ																																																			
3	31:27	Surface Format Format: U5 Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.																																																			
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>YCRCB_NORMAL</td> <td></td> </tr> <tr> <td>1</td> <td>YCRCB_SWAPUVY</td> <td></td> </tr> <tr> <td>2</td> <td>YCRCB_SWAPUV</td> <td></td> </tr> <tr> <td>3</td> <td>YCRCB_SWAPY</td> <td></td> </tr> <tr> <td>4</td> <td>PLANAR_420_8</td> <td>NV12 with Interleave Chroma set</td> </tr> <tr> <td>5</td> <td>PACKED_444A_8</td> <td></td> </tr> <tr> <td>6</td> <td>PACKED_422_16</td> <td></td> </tr> <tr> <td>7</td> <td>R10G10B10A2_UNORM / R10G10B10A2_UNORM_SRGB</td> <td></td> </tr> <tr> <td>8</td> <td>R8G8B8A8_UNORM / R8G8B8A8_UNORM_SRGB</td> <td></td> </tr> <tr> <td>9</td> <td>PACKED_444_16</td> <td></td> </tr> <tr> <td>10</td> <td>PLANAR_422_16</td> <td></td> </tr> <tr> <td>11</td> <td>Y8_UNORM</td> <td></td> </tr> <tr> <td>12</td> <td>PLANAR_420_16</td> <td></td> </tr> <tr> <td>13</td> <td>R16G16B16A16</td> <td></td> </tr> <tr> <td>14</td> <td>Bayer pattern</td> <td></td> </tr> <tr> <td>15</td> <td>Y16_UNORM</td> <td></td> </tr> </tbody> </table>	Value	Name	Description	0	YCRCB_NORMAL		1	YCRCB_SWAPUVY		2	YCRCB_SWAPUV		3	YCRCB_SWAPY		4	PLANAR_420_8	NV12 with Interleave Chroma set	5	PACKED_444A_8		6	PACKED_422_16		7	R10G10B10A2_UNORM / R10G10B10A2_UNORM_SRGB		8	R8G8B8A8_UNORM / R8G8B8A8_UNORM_SRGB		9	PACKED_444_16		10	PLANAR_422_16		11	Y8_UNORM		12	PLANAR_420_16		13	R16G16B16A16		14	Bayer pattern		15	Y16_UNORM	
		Value	Name	Description																																																	
		0	YCRCB_NORMAL																																																		
		1	YCRCB_SWAPUVY																																																		
		2	YCRCB_SWAPUV																																																		
		3	YCRCB_SWAPY																																																		
		4	PLANAR_420_8	NV12 with Interleave Chroma set																																																	
		5	PACKED_444A_8																																																		
		6	PACKED_422_16																																																		
		7	R10G10B10A2_UNORM / R10G10B10A2_UNORM_SRGB																																																		
		8	R8G8B8A8_UNORM / R8G8B8A8_UNORM_SRGB																																																		
		9	PACKED_444_16																																																		
		10	PLANAR_422_16																																																		
		11	Y8_UNORM																																																		
12	PLANAR_420_16																																																				
13	R16G16B16A16																																																				
14	Bayer pattern																																																				
15	Y16_UNORM																																																				
26:25	Bayer Pattern Offset	Specifies the starting pixel offset for the Bayer pattern used for Capture Pipe.																																																			
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Pixel at X=0, Y=0 is Blue</td> </tr> <tr> <td>01b</td> <td>Pixel at X=0, Y=0 is Red</td> </tr> <tr> <td>10b</td> <td>Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Red</td> </tr> <tr> <td>11b</td> <td>Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Blue</td> </tr> </tbody> </table>	Value	Name	00b	Pixel at X=0, Y=0 is Blue	01b	Pixel at X=0, Y=0 is Red	10b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Red	11b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Blue																																									
		Value	Name																																																		
		00b	Pixel at X=0, Y=0 is Blue																																																		
		01b	Pixel at X=0, Y=0 is Red																																																		
10b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Red																																																				
11b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Blue																																																				
24	Bayer Pattern Format Specifies the format of the Bayer Pattern:																																																				
<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>8-bit input at a 8-bit stride</td> </tr> <tr> <td>1b</td> <td>16-bit input at a 16-bit stride</td> </tr> </tbody> </table>	Value	Name	0b	8-bit input at a 8-bit stride	1b	16-bit input at a 16-bit stride																																															
Value	Name																																																				
0b	8-bit input at a 8-bit stride																																																				
1b	16-bit input at a 16-bit stride																																																				
23:22	Bayer Input Alignment																																																				

VEBOX_SURFACE_STATE		
	Format:	U2
	Value	Name
	00b	MSB aligned data [Default]
	01b	10bit LSB aligned data
	10b	12bit LSB aligned data
	11b	14bit LSB aligned data
	Programming Notes	
	Valid only Bayer Pattern Format is 16bit input	
21	Reserved	
	Format:	MBZ
20	Interleave Chroma	
	Format:	Enable
	This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats.	
19:3	Surface Pitch	
	Format:	U17
	This field specifies the surface pitch in (#Bytes - 1):	
	Value	Name
	[63, 131071]	For other linear surfaces [64B, 128KB]
	[511, 131071]	For X-tiled surface [512B, 128KB] = [1tile, 256 tiles]
	[127, 131071]	For Y-tiled surfaces [128B,128KB] = [1 tile, 1024 tiles]
	Programming Notes	
	For tiled surfaces, the pitch must be a multiple of the tile width.For linear surfaces, the pitch must be a multiple of 64.If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.	
2	Half Pitch for Chroma	
	Format:	Enable
	This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats.	
	Programming Notes	
	Must be programmed to Zero always as this field is not used	
1	Tiled Surface	
	Format:	Boolean
	This field specifies whether the surface is tiled.	
	Value	Name
		Description

VEBOX_SURFACE_STATE										
		<table border="1"> <tr> <td>1</td> <td>True</td> <td>Tiled</td> </tr> <tr> <td>0</td> <td>False</td> <td>Linear</td> </tr> </table>	1	True	Tiled	0	False	Linear		
1	True	Tiled								
0	False	Linear								
		<p style="text-align: center;">Programming Notes</p> <p>Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory. The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.</p>								
	0	<p>Tile Walk</p> <table border="1"> <tr> <td>Format:</td> <td>Boolean</td> </tr> </table> <p>This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See <i>Memory Interface Functions</i> for details on memory tiling and restrictions. This field is ignored when the surface is linear.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>TILEWALK_XMAJOR</td> </tr> <tr> <td>1</td> <td>TILEWALK_YMAJOR</td> </tr> </tbody> </table> <p style="text-align: center;">Programming Notes</p> <p>The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.</p>	Format:	Boolean	Value	Name	0	TILEWALK_XMAJOR	1	TILEWALK_YMAJOR
Format:	Boolean									
Value	Name									
0	TILEWALK_XMAJOR									
1	TILEWALK_YMAJOR									
4	31:29	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									
	28:16	<p>X Offset for U</p> <table border="1"> <tr> <td>Format:</td> <td>U13</td> </tr> </table> <p>This field must be zero for the VEBOX surface formats</p>	Format:	U13						
Format:	U13									
	15	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									
	14:0	<p>Y Offset for U</p> <table border="1"> <tr> <td>Format:</td> <td>U15</td> </tr> </table> <p>This field specifies the vertical offset in rows from the start (origin) or the Luma(Y) plane?to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats.</p> <p style="text-align: center;">Programming Notes</p> <p>This field must indicate an even number (bit 0 = 0).This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line) For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for U should be an integral multiple of the Tile height of the Luma plane</p>	Format:	U15						
Format:	U15									
5	31:29	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									

VEBOX_SURFACE_STATE					
	28:16	X Offset for V Format: U13 This field must be zero for the VEBOX surface formats.			
	15	Reserved Format: MBZ			
	14:0	Y Offset for V Format: U15 This field specifies the vertical offset in rows from the start (origin) of the Luma(Y) plane to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. <table border="1" style="width: 100%; margin-top: 5px;"> <tr> <th style="text-align: center; color: #0070C0;">Programming Notes</th> </tr> <tr> <td style="padding: 2px;">This field must indicate an even number (bit 0 = 0). This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line). For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for V should be an integral multiple of the Tile height of the Luma plane</td> </tr> </table>	Programming Notes	This field must indicate an even number (bit 0 = 0). This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line). For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for V should be an integral multiple of the Tile height of the Luma plane	
Programming Notes					
This field must indicate an even number (bit 0 = 0). This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line). For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for V should be an integral multiple of the Tile height of the Luma plane					
6	31	Reserved Format: MBZ			
	30:16	X Offset for Frame Format: U15 This is an offset in X from the Surface Base Address in pixels for all planes using this surface. For U/V planes this is added to the X Offset for U/V. After converting to bytes this must be an integer multiple of cache lines in the tiling mode. This specifies the edge of the frame, so adjacent pixels needed for Denoise/Deinterlace/Demosaic are replicated or mirrored. <table border="1" style="width: 100%; margin-top: 5px;"> <tr> <th style="text-align: center; color: #0070C0;">Programming Notes</th> </tr> <tr> <td style="padding: 2px;">If Y Offset for Frame >0 the X Offset must be 0.</td> </tr> <tr> <td style="padding: 2px;">If memory compression is enabled then this must be an even number of cache lines.</td> </tr> </table>	Programming Notes	If Y Offset for Frame >0 the X Offset must be 0.	If memory compression is enabled then this must be an even number of cache lines.
	Programming Notes				
	If Y Offset for Frame >0 the X Offset must be 0.				
	If memory compression is enabled then this must be an even number of cache lines.				
15	Reserved Format: MBZ				
14:0	Y Offset for Frame Format: U15 This is an offset in Y from the Surface Base Address in pixels for all planes using this surface. For U/V planes this is added to the Y Offset for U/V. After converting to bytes this must be an integer multiple of cache lines in the tiling mode. This specifies the edge of the frame, so adjacent pixels needed for Denoise/Deinterlace/Demosaic are replicated or mirrored. <table border="1" style="width: 100%; margin-top: 5px;"> <tr> <th style="text-align: center; color: #0070C0;">Programming Notes</th> </tr> <tr> <td style="padding: 2px;">If X Offset for Frame >0 the Y Offset must be 0. For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for Frame should be an integral multiple of the Tile height.</td> </tr> </table>	Programming Notes	If X Offset for Frame >0 the Y Offset must be 0. For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for Frame should be an integral multiple of the Tile height.		
Programming Notes					
If X Offset for Frame >0 the Y Offset must be 0. For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for Frame should be an integral multiple of the Tile height.					
7	31:27	Reserved Format: MBZ			

VEBOX_SURFACE_STATE

	26:17	Reserved	Format: MBZ																	
	16:0	Derived Surface Pitch	Format: U17																	
			<p>This field specifies the surface pitch in (#Bytes - 1) for the derived surfaces: STMM/Denoise statistic surface is described when the Surface Identification bit is 0 (Input Surface). The (Current Denoise Output)/(Previous Denoise Input) surfaces are described when the bit is 1 (Output Surface).</p>																	
			<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 35%;">Description</th> <th style="width: 35%;">Exists If</th> </tr> </thead> <tbody> <tr> <td>[63, 131071]</td> <td></td> <td>[64B, 128KB]</td> <td>[Tiled Surface] == 0</td> </tr> <tr> <td>[511, 131071]</td> <td></td> <td>[512B, 128KB] = [1tile, 256 tiles]</td> <td>([Tiled Surface] == 1) AND ([Tile Walk] == 0)</td> </tr> <tr> <td>[127, 131071]</td> <td></td> <td>[128B, 128KB] = [1 tile, 1024 tiles]</td> <td>([Tiled Surface] == 1) AND ([Tile Walk] == 1)</td> </tr> </tbody> </table>		Value	Name	Description	Exists If	[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0	[511, 131071]		[512B, 128KB] = [1tile, 256 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 0)	[127, 131071]		[128B, 128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)
Value	Name	Description	Exists If																	
[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0																	
[511, 131071]		[512B, 128KB] = [1tile, 256 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 0)																	
[127, 131071]		[128B, 128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)																	
			Programming Notes																	
			In DN Only mode, the pitch for the (Current Denoise Output)/(Previous Denoise Input) and the Surface Pitch must be programed the same.																	
			The pitch must be a multiple of the tile width.																	
8	31:17	Reserved	Format: MBZ																	
	16:0	Surface Pitch for Skin Score Output Surfaces	Format: U17																	
			<p>This field specifies the surface pitch in (#Bytes - 1) for the Skin Score Output surface if enabled; This is present only in the output surface format and reserved for Input surface format. The height and width are the same as in the Output surface mentioned above.</p>																	
			<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr style="background-color: #e1eef6;"> <th style="width: 15%;">Value</th> <th style="width: 15%;">Name</th> <th style="width: 35%;">Description</th> <th style="width: 35%;">Exists If</th> </tr> </thead> <tbody> <tr> <td>[63, 131071]</td> <td></td> <td>[64B, 128KB]</td> <td>[Tiled Surface] == 0</td> </tr> <tr> <td>[511, 131071]</td> <td></td> <td>[512B, 128KB] = [1tile, 256 tiles]</td> <td>([Tiled Surface] == 1) AND ([Tile Walk] == 0)</td> </tr> <tr> <td>[127, 131071]</td> <td></td> <td>[128B, 128KB] = [1 tile, 1024 tiles]</td> <td>([Tiled Surface] == 1) AND ([Tile Walk] == 1)</td> </tr> </tbody> </table>		Value	Name	Description	Exists If	[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0	[511, 131071]		[512B, 128KB] = [1tile, 256 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 0)	[127, 131071]		[128B, 128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)
Value	Name	Description	Exists If																	
[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0																	
[511, 131071]		[512B, 128KB] = [1tile, 256 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 0)																	
[127, 131071]		[128B, 128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)																	
			Programming Notes																	
			The pitch must be a multiple of the tile width.																	

VEBOX_TILING_CONVERT

VEBOX_TILING_CONVERT								
Source:	VideoEnhancementCS							
Length Bias:	2							
<p>This command takes the input surface and writes directly to the output surface at high speed. The surface format and width/height of the input and output must be the same, only the tiling mode and pitch can change.</p>								
DWord	Bit	Description						
0	31:29	Command Type						
		Default Value:	3h PARALLEL_VIDEO_PIPE					
		Format:	OpCode					
	28:27	Pipeline						
		Default Value:	2h Media					
		Format:	OpCode					
	26:24	Command OpCode						
		Default Value:	4h VEBOX					
		Format:	OpCode					
	23:21	SubOpcode A						
Default Value:		0h						
Format:		OpCode						
20:16	SubOpcode B							
	Default Value:	1h						
	Format:	OpCode						
15:12	Reserved							
	Format:	MBZ						
11:0	11:0	DWord Length						
		Format:	=n					
	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>3h</td> <td></td> <td>(Excludes DWords 0, 1)</td> </tr> </tbody> </table>		Value	Name	Description	3h		(Excludes DWords 0, 1)
	Value	Name	Description					
3h		(Excludes DWords 0, 1)						
1..2	63:12	Input Address						
		Format:	VIRTUAL_ADDR[63:12]					
	<p>Specifies bits 47:12 of the 4Kbyte-aligned frame buffer address for reading the current frame.</p>							
11	Reserved							
	Format:	MBZ						
	10:0	Input Surface Control Bits						

VEBOX_TILING_CONVERT		
		Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS
3.4 Programming Notes: Output address must be different from input address	63:12	Output Address Format: VIRTUAL_ADDR[63:12] Specifies bits 47:12 of the 4Kbyte-aligned frame buffer address for writing the current frame.
	11	Reserved Format: MBZ
	10:0	Output Surface Control Bits Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS



Wait for Event

MSD_WAIT_FOR_EVENT - Wait for Event							
Source:	EuSubFunctionGateway						
Length Bias:	1						
Send a writeback if Event ID occurred after MonitorEvent.							
DWord	Bit	Description					
0	31:29	Reserved					
		Format: MBZ					
	28:25	Message Length					
		Format: U4					
		Specifies the number of GRF registers sent as the message payload.					
		<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>One [Default]</td> <td>See MDP_TIMEOUT Timeout Data Payload definition.</td> </tr> </tbody> </table>	Value	Name	Description	1	One [Default]
	Value	Name	Description				
	1	One [Default]	See MDP_TIMEOUT Timeout Data Payload definition.				
	24:20	Response Length					
		Format: U5					
Specifies the number of GRF registers expected as the message response payload.							
<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>One [Default]</td> <td>Ignore the data in the register. The update of the register is used as the event completion notification</td> </tr> </tbody> </table>		Value	Name	Description	1	One [Default]	Ignore the data in the register. The update of the register is used as the event completion notification
Value	Name	Description					
1	One [Default]	Ignore the data in the register. The update of the register is used as the event completion notification					
19:3	Reserved						
	Format: MBZ						
2:0	Wait for Event Subfunction						
	Default Value: 0x6						
	Format: OpCode						

While

while - While			
Source:	Eulsa		
Length Bias:	4		
<p>The while instruction marks the end of a do-while block. The instruction first evaluates the loop termination condition for each channel based on the current channel enables and the predication flags specified in the instruction. If any channel has not terminated, a branch is taken to a destination address specified in the instruction, and the loop continues for those channels. Otherwise, execution continues to the next instruction. It should be a negative number for the backward referencing. If SPF is ON, none of the PclP are updated.</p>			
Format:	<code>[(pred)] while (exec_size) JIP</code>		
Restriction			
<p>The execution size must be the same for the while instruction and any break and cont instructions of the same code block.</p>			
Syntax			
<code>[(pred)] while (exec_size) imm32</code>			
Pseudocode			
<pre> Evaluate(WrEn); for (n = 0; n < 32; n++) { if (WrEn.chan[n]) { PcIP[n] = IP + JIP; } else { PcIP[n] = IP + 1; } } if (PMask == 1) { // any enabled channel true Jump(IP + JIP); } </pre>			
Predication	Conditional Modifier	Saturation	Source Modifier
Y	N	N	N
DWord	Bit	Description	
0..3	127:96	Reserved	
		Exists If:	<code>([Src0.IsImm]==false)</code>
	Format:	MBZ	
	127:96	JIP	
		Exists If:	<code>([Src0.IsImm]==true)</code>
		Format:	S31
The byte-aligned jump distance if a jump is taken for the channel			
95:80	Reserved		
	Exists If:	<code>([Src0.IsImm]==false)</code>	
	Format:	MBZ	

while - While

	while - While		
95:64	Reserved		
	Exists If:	([Src0.IsImm]==true)	
	Format:	MBZ	
79:66	Src0.Operand		
	Exists If:	([Src0.IsImm]==false)	
	Format:	DirectOperand	
65:64	Reserved		
	Exists If:	([Src0.IsImm]==false)	
	Format:	MBZ	
63:50	Dst.Operand		
	Format:	DirectOperand	
49:47	Reserved		
	Format:	MBZ	
46	Src0.IsImm	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	Name	
	0	false	
	1	true	
45:34	Reserved		
	Format:	MBZ	
33	BranchCtrl	This field is used by <i>goto</i> , <i>if</i> , and <i>else</i> instructions to control branching. See the goto instruction description for more information about BranchCtrl.	
32	AtomicCtrl		
	Format:	AtomicCtrl	
31	MaskCtrl	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	Name	Description
	0	Normal [Default]	Normal. Per channel write enable used for final write enable generation.
	1	NoMask	NoMask. Skips the check for PclP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	Reserved		
29	CmptCtrl		
	Format:	MBZ	
	Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU		

while - While

	<p>decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NoCompaction [Default]</td> <td>No compaction. 128-bit native instruction supporting all instruction options.</td> </tr> <tr> <td>1</td> <td>Compacted</td> <td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td> </tr> </tbody> </table>	Value	Name	Description	0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.	1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
Value	Name	Description								
0	NoCompaction [Default]	No compaction. 128-bit native instruction supporting all instruction options.								
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.								
28	<p>PredInv This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Positive [Default]</td> <td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td> </tr> <tr> <td>1</td> <td>Negative</td> <td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td> </tr> </tbody> </table>	Value	Name	Description	0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.	1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
Value	Name	Description								
0	Positive [Default]	Positive polarity of predication. Use the predication mask produced by PredCtrl.								
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.								
27:24	<p>PredCtrl</p> <table border="1"> <tr> <td>Format:</td> <td>PredCtrl</td> </tr> </table> <p>This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</p>	Format:	PredCtrl							
Format:	PredCtrl									
23	<p>FlagRegNum[0] This field specifies bit[0] of the register number for a flag register operand.</p>									
22	<p>FlagSubRegNum This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</p>									
21:19	<p>ChanOff</p> <table border="1"> <tr> <td>Format:</td> <td>ChanOff</td> </tr> </table> <p>This field provides offset information for ARF selection. This can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</p>	Format:	ChanOff							
Format:	ChanOff									

while - While	
18:16	ExecSize Format: ExecSize This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	Header Format: Header

Word Atomic Counter with Return Data Operation MSD

MSD1R_WAC - Word Atomic Counter with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header
	18:14	Message Type

MSD1R_WAC - Word Atomic Counter with Return Data Operation MSD

		Default Value:	0Ch
		Format:	Opcode
		Atomic Half Counter Operation message	
	13	Return Data Control	
		Default Value:	1h
		Format:	Opcode
		Specifies that return data is sent back to the thread.	
	12	Reserved	
		Format:	MBZ
	11:8	Atomic Integer Operation	
		Format:	MDC_AOP
		Specifies the atomic integer operation to be performed.	
	7:0	Binding Table Index	
		Format:	MDC_BTS
		Specifies the Binding Table Index for the message	

Word Atomic Counter Write Only Operation MSD

MSD1W_WAC - Word Atomic Counter Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHR Indicates that the message requires a header
	18:14	Message Type Default Value: 0Ch

MSD1W_WAC - Word Atomic Counter Write Only Operation MSD

		Format:	Opcode
		Atomic Half Counter Operation message	
13	Return Data Control		
		Default Value:	0h
		Format:	Opcode
		Specifies that no return data is sent back to the thread.	
12	Reserved		
		Format:	MBZ
11:8	Atomic Integer Operation		
		Format:	MDC_AOP
		Specifies the atomic integer operation to be performed.	
7:0	Binding Table Index		
		Format:	MDC_BTS
		Specifies the Binding Table Index for the message	

Word Typed Atomic Integer with Return Data Operation MSD

MSD1R_WTAI - Word Typed Atomic Integer with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type

MSD1R_WTAI - Word Typed Atomic Integer with Return Data Operation MSD

		Default Value:	07h
		Format:	Opcode
		Typed Atomic Half Integer Operation message	
	13	Return Data Control	
		Default Value:	1h
		Format:	Opcode
		Specifies that return data is sent back to the thread.	
	12	Reserved	
		Format:	MBZ
	11:8	Atomic Integer Operation	
		Format:	MDC_AOP
		Specifies the atomic integer operation to be performed.	
	7:0	Binding Table Index	
		Format:	MDC_BTS
		Specifies the Binding Table Index for the message	

Word Typed Atomic Integer Write Only Operation MSD

MSD1W_WTAI - Word Typed Atomic Integer Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type

MSD1W_WTAI - Word Typed Atomic Integer Write Only Operation MSD

		Default Value:	07h
		Format:	Opcode
	Typed Atomic Half Integer Operation message		
	13	Return Data Control	
		Default Value:	0h
		Format:	Opcode
	Specifies that no return data is sent back to the thread.		
	12	Reserved	
		Format:	MBZ
	11:8	Atomic Integer Operation	
		Format:	MDC_AOP
	Specifies the atomic integer operation to be performed.		
	7:0	Binding Table Index	
	Format:	MDC_BTS	
Specifies the Binding Table Index for the message			

Word Untyped Atomic Float with Return Data Operation MSD

MSD1R_WAF - Word Untyped Atomic Float with Return Data Operation MSD										
Source:		EuSubFunctionDataPort1								
Length Bias:		1								
DWord	Bit	Description								
0	31	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
	30	Packed Data Payload <table border="1"> <tr> <td>Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF).</p> <table border="1"> <tr> <th colspan="2">Restriction</th> </tr> <tr> <td colspan="2">Only 32-bit data packing is supported at this time.</td> </tr> </table>	Default Value:	0 32 bit	Format:	Enable	Restriction		Only 32-bit data packing is supported at this time.	
	Default Value:	0 32 bit								
	Format:	Enable								
	Restriction									
	Only 32-bit data packing is supported at this time.									
	29	Packed Address Payload <table border="1"> <tr> <td>Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.</p>	Default Value:	0 32 bit	Format:	Enable				
	Default Value:	0 32 bit								
	Format:	Enable								
28:25	Message Length <table border="1"> <tr> <td>Format:</td> <td>U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4							
Format:	U4									
24:20	Response Length <table border="1"> <tr> <td>Format:</td> <td>U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5							
Format:	U5									
19	Header Present <table border="1"> <tr> <td>Format:</td> <td>MDC_MHP</td> </tr> </table> <p>If set, indicates that the message includes the header.</p>	Format:	MDC_MHP							
Format:	MDC_MHP									
18:14	Message Type									

MSD1R_WAF - Word Untyped Atomic Float with Return Data Operation MSD

		Default Value:	1Ch
		Format:	Opcode
	Untyped Atomic Half Float Operation message		
13	Return Data Control		
		Default Value:	1h
		Format:	Opcode
	Specifies that return data is sent back to the thread.		
12	SIMD Mode		
	Format:	MDC_SM2R	
	Specifies the SIMD mode of the message (number of slots processed)		
11	Reserved		
	Format:	MBZ	
10:8	Atomic Float Operation		
	Format:	MDC_FOP	
	Specifies the atomic float operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS_SLM_A32	
	Specifies the Binding Table Index for the message		

Word Untyped Atomic Float Write Only Operation MSD

MSD1W_WAF - Word Untyped Atomic Float Write Only Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type

MSD1W_WAF - Word Untyped Atomic Float Write Only Operation MSD

		Default Value:	1Ch
		Format:	Opcode
	Untyped Atomic Half Float Operation message		
13	Return Data Control		
		Default Value:	0h
		Format:	Opcode
	Specifies that no return data is sent back to the thread.		
12	SIMD Mode		
	Format:	MDC_SM2R	
	Specifies the SIMD mode of the message (number of slots processed)		
11	Reserved		
	Format:	MBZ	
10:8	Atomic Float Operation		
	Format:	MDC_FOP	
	Specifies the atomic float operation to be performed.		
7:0	Binding Table Index		
	Format:	MDC_BTS_SLM_A32	
	Specifies the Binding Table Index for the message		

Word Untyped Atomic Integer with Return Data Operation MSD

MSD1R_WAI - Word Untyped Atomic Integer with Return Data Operation MSD		
Source:		EuSubFunctionDataPort1
Length Bias:		1
DWord	Bit	Description
0	31	Reserved Format: MBZ
	30	Packed Data Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). Restriction Only 32-bit data packing is supported at this time.
	29	Packed Address Payload Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	Message Length Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	Response Length Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	Header Present Format: MDC_MHP If set, indicates that the message includes the header.
	18:14	Message Type

MSD1R_WAI - Word Untyped Atomic Integer with Return Data Operation MSD

		Default Value:	03h
		Format:	Opcode
	Untyped Atomic Half Integer Operation message		
	13	Return Data Control	
		Default Value:	1h
		Format:	Opcode
	Specifies that return data is sent back to the thread.		
	12	SIMD Mode	
		Format:	MDC_SM2R
	Specifies the SIMD mode of the message (number of slots processed)		
	11:8	Atomic Integer Operation	
		Format:	MDC_AOP
	Specifies the atomic integer operation to be performed.		
	7:0	Binding Table Index	
		Format:	MDC_BTS_SLM_A32
	Specifies the Binding Table Index for the message		

Word Untyped Atomic Integer Write Only Operation MSD

MSD1W_WAI - Word Untyped Atomic Integer Write Only Operation MSD										
Source:		EuSubFunctionDataPort1								
Length Bias:		1								
DWord	Bit	Description								
0	31	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
	30	Packed Data Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF).</p> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th colspan="2" style="text-align: center;">Restriction</th> </tr> <tr> <td colspan="2" style="text-align: center;">Only 32-bit data packing is supported at this time.</td> </tr> </table>	Default Value:	0 32 bit	Format:	Enable	Restriction		Only 32-bit data packing is supported at this time.	
	Default Value:	0 32 bit								
	Format:	Enable								
	Restriction									
	Only 32-bit data packing is supported at this time.									
	29	Packed Address Payload <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Default Value:</td> <td>0 32 bit</td> </tr> <tr> <td>Format:</td> <td>Enable</td> </tr> </table> <p>When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.</p>	Default Value:	0 32 bit	Format:	Enable				
Default Value:	0 32 bit									
Format:	Enable									
28:25	Message Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U4</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.</p>	Format:	U4							
Format:	U4									
24:20	Response Length <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>U5</td> </tr> </table> <p>Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.</p>	Format:	U5							
Format:	U5									
19	Header Present <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="color: red;">MDC_MHP</td> </tr> </table> <p>If set, indicates that the message includes the header.</p>	Format:	MDC_MHP							
Format:	MDC_MHP									
18:14	Message Type									

MSD1W_WAI - Word Untyped Atomic Integer Write Only Operation MSD

	<table border="1"> <tr> <td>Default Value:</td> <td>03h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table> <p>Untyped Atomic Half Integer Operation message</p>	Default Value:	03h	Format:	Opcode
Default Value:	03h				
Format:	Opcode				
13	<p>Return Data Control</p> <table border="1"> <tr> <td>Default Value:</td> <td>0h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table> <p>Specifies that no return data is sent back to the thread.</p>	Default Value:	0h	Format:	Opcode
Default Value:	0h				
Format:	Opcode				
12	<p>SIMD Mode</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_SM2R</td> </tr> </table> <p>Specifies the SIMD mode of the message (number of slots processed)</p>	Format:	MDC_SM2R		
Format:	MDC_SM2R				
11:8	<p>Atomic Integer Operation</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_AOP</td> </tr> </table> <p>Specifies the atomic integer operation to be performed.</p>	Format:	MDC_AOP		
Format:	MDC_AOP				
7:0	<p>Binding Table Index</p> <table border="1"> <tr> <td>Format:</td> <td>MDC_BTS_SLM_A32</td> </tr> </table> <p>Specifies the Binding Table Index for the message</p>	Format:	MDC_BTS_SLM_A32		
Format:	MDC_BTS_SLM_A32				

XY_BLOCK_COPY_BLT

XY_BLOCK_COPY_BLT			
Source:	BlitterCS		
Length Bias:	2		
Description			
<p>XY_BLOCK_COPY_BLT instruction performs a color source copy where the only operands involved are a color source and destination of the same bit width. The source and destination surfaces CAN overlap, the hardware handles this internally. This new blit command will happen in large numbers, consecutively, possibly an entire batch will comprise only new blit commands. Legacy commands and new blit command will not be interspersed. If they are, they will be separated by implied HW flush: Whenever there is a transition between this new Fast Blit command and the Legacy Blit commands (2D BLT instructions other than XY_BLOCK_COPY_BLT, XY_FAST_COPY_BLT, XY_FAST_COLOR_BLT), the HW will impose an automatic flush BEFORE the execution (at the beginning) of the next blitter command. The starting pixel of the blit operation for both source and destination should be on a pixel boundary.</p> <p>Note that when two sequential block copy blits have different source surfaces, but their destinations refer to the same destination surfaces and therefore destinations overlap it is imperative that a Flush be inserted between the two blits.</p>			
DWord	Bit	Description	
0	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Format:	Opcode
		Value	Name
		41h	[Default]
	21:19	Color Depth	
		This field actually programs bits per pixel value for each pixel of the surface. Reprogramming of these bits require explicit flushing of Copy Engine.	
		Value	Name
000b		8 bit color [Default]	
001b		16 bit color	
010b		32 bit color	
011b		64 bit color	
100b		96 bit color (only linear case is supported)	
101b		128 bit color	
110b		RESERVED	
111b	RESERVED		

XY_BLOCK_COPY_BLT												
		<table border="1" style="width: 100%;"> <tr> <th colspan="2" style="text-align: center;">Programming Notes</th> </tr> <tr> <td colspan="2">Color depth programming for 96 bit color is invalid unless both the source and destination surfaces are linear.</td> </tr> </table>	Programming Notes		Color depth programming for 96 bit color is invalid unless both the source and destination surfaces are linear.							
Programming Notes												
Color depth programming for 96 bit color is invalid unless both the source and destination surfaces are linear.												
	18:14	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	13:9	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	8	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	7:0	DWord Length <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>=n</td> </tr> </table> <table border="1" style="width: 100%; margin-top: 5px;"> <tr> <th colspan="2" style="text-align: center;">Description</th> </tr> <tr> <td colspan="2">n = 10 This field indicates length of the instruction in DWORD.</td> </tr> <tr> <th style="width: 30%;">Value</th> <th>Name</th> </tr> <tr> <td>10</td> <td>Excludes DWORD 0,1 [Default]</td> </tr> </table>	Format:	=n	Description		n = 10 This field indicates length of the instruction in DWORD.		Value	Name	10	Excludes DWORD 0,1 [Default]
Format:	=n											
Description												
n = 10 This field indicates length of the instruction in DWORD.												
Value	Name											
10	Excludes DWORD 0,1 [Default]											
1	31:30	Destination Tiling These bits indicate destination tiling method. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 40%;">Description</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>LINEAR [Default]</td> <td>Linear mode (no tiling)</td> </tr> <tr> <td>01b</td> <td>YMAJOR</td> <td>Y major tiling</td> </tr> </tbody> </table>	Value	Name	Description	00b	LINEAR [Default]	Linear mode (no tiling)	01b	YMAJOR	Y major tiling	
Value	Name	Description										
00b	LINEAR [Default]	Linear mode (no tiling)										
01b	YMAJOR	Y major tiling										
	29:28	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	27:21	Destination MOCS value MOCS (Memory Object State Control) for destination operand.										
	20:18	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	17:0	Destination Pitch <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>U18-1</td> </tr> </table> <p>For Linear Surfaces, the pitch must be multiple of pixel width in bytes. For Tiled surfaces, the pitch has to be a multiple of the Tile Width (X direction width of the Tile) expressed in dwords (4 bytes).</p>	Format:	U18-1								
Format:	U18-1											
2	31:16	Destination Y1 Coordinate (Top) 16-bit signed number. The destination start line (inclusive) for Block Copy blit.										
	15:0	Destination X1 Coordinate (Left)										

XY_BLOCK_COPY_BLT					
		16-bit signed number. The destination start pixel (inclusive) for Block Copy blit.			
3	31:16	Destination Y2 Coordinate (Bottom) 16-bit signed number. The destination end line (exclusive) for Block Copy blit.			
	15:0	Destination X2 Coordinate (Right) 16-bit signed number. The destination end pixel (exclusive) for Block Copy blit.			
4..5	63:0	Destination Base Address			
		<table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>GraphicsAddress[63:0]</td> </tr> </table> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Description</th> </tr> <tr> <td>This bitfield contains the base address of the destination surface. When Tiling is enabled this address is cacheline (64Byte) aligned. When Destination Tiling is disabled, this address is byte aligned.</td> </tr> </table>	Format:	GraphicsAddress[63:0]	Description
Format:	GraphicsAddress[63:0]				
Description					
This bitfield contains the base address of the destination surface. When Tiling is enabled this address is cacheline (64Byte) aligned. When Destination Tiling is disabled, this address is byte aligned.					
6	31	Reserved			
		<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
	Format:	MBZ			
	30	Reserved			
		<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ	
	Format:	MBZ			
29:16	Destination Y offset				
	<table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Description</th> </tr> <tr> <td>Format is U14. This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface.</td> </tr> </table> <table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> <tr> <td>For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.</td> </tr> </table>	Description	Format is U14. This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface.	Programming Notes	For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.
Description					
Format is U14. This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface.					
Programming Notes					
For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.					
15:14	Reserved				
	<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ		
Format:	MBZ				
7	13:0	Destination X offset			
		<table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Description</th> </tr> <tr> <td>Format is U14. This field specifies the horizontal offset in pixels from the surface base address to the start (origin) of the surface.</td> </tr> </table>	Description	Format is U14. This field specifies the horizontal offset in pixels from the surface base address to the start (origin) of the surface.	
		Description			
Format is U14. This field specifies the horizontal offset in pixels from the surface base address to the start (origin) of the surface.					
<table border="1" style="width: 100%; background-color: #e6f2ff;"> <tr> <th style="text-align: center;">Programming Notes</th> </tr> <tr> <td>For Linear surface X offset must be 0. For Tiled surface X offset must be greater than or equal to 0 and less than 16K in pixels.</td> </tr> </table>	Programming Notes	For Linear surface X offset must be 0. For Tiled surface X offset must be greater than or equal to 0 and less than 16K in pixels.			
Programming Notes					
For Linear surface X offset must be 0. For Tiled surface X offset must be greater than or equal to 0 and less than 16K in pixels.					
7	31:16	Source Y1 Coordinate (Top) Format is S16.			

XY_BLOCK_COPY_BLT																	
		The source start line (inclusive) for the Block Copy blit.															
	15:0	Source X1 Coordinate (Left) Format is S16. Source start pixel (inclusive) for Block Copy blit.															
8	31:30	Source Tiling <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="3" style="text-align: center; background-color: #e6f2ff;">Description</th> </tr> </thead> <tbody> <tr> <td colspan="3">These bits indicate source tiling method.</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> <tr> <td style="text-align: center;">00b</td> <td style="text-align: center;">LINEAR [Default]</td> <td>Linear Tiling (tiking disabled)</td> </tr> <tr> <td style="text-align: center;">01b</td> <td style="text-align: center;">YMAJOR</td> <td>Y major tiling</td> </tr> </tbody> </table>	Description			These bits indicate source tiling method.			Value	Name	Description	00b	LINEAR [Default]	Linear Tiling (tiking disabled)	01b	YMAJOR	Y major tiling
		Description															
		These bits indicate source tiling method.															
		Value	Name	Description													
	00b	LINEAR [Default]	Linear Tiling (tiking disabled)														
	01b	YMAJOR	Y major tiling														
	29:28	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%; text-align: center;">MBZ</td></tr></table>		MBZ													
		MBZ															
27:21	Source MOCS <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="3" style="text-align: center; background-color: #e6f2ff;">Description</th> </tr> </thead> <tbody> <tr> <td colspan="3">MOCS (Memory Object State Control) value for source operand.</td> </tr> </tbody> </table>	Description			MOCS (Memory Object State Control) value for source operand.												
Description																	
MOCS (Memory Object State Control) value for source operand.																	
20:18	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%; text-align: center;">MBZ</td></tr></table>		MBZ														
	MBZ																
17:0	Source Pitch Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%; text-align: center;">U18-1</td></tr></table> For Linear surfaces , the pitch must be multiple of pixel width in bytes. For tiled surfaces, the pitch has to be multiple of the Tile width (X direction width of the tile) expressed in dwords (4 byte).		U18-1														
	U18-1																
9..10	Source Base Address Format: <table border="1" style="width: 100%;"><tr><td style="width: 40%;"></td><td style="width: 60%; text-align: center;">GraphicsAddress[63:0]</td></tr></table> This bitfield contains the base address of the source surface. When Tiling is enabled this address is cacheline (64Byte) aligned. When Source Tiling is disabled, this address is byte aligned.		GraphicsAddress[63:0]														
	GraphicsAddress[63:0]																
11	31	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%; text-align: center;">MBZ</td></tr></table>		MBZ													
		MBZ															
	30	Reserved Format: <table border="1" style="width: 100%;"><tr><td style="width: 80%;"></td><td style="width: 20%; text-align: center;">MBZ</td></tr></table>		MBZ													
	MBZ																
29:16	Source Y offset Format U14.																

XY_BLOCK_COPY_BLT		
		<p>This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface.</p> <p style="text-align: center;">Programming Notes</p> <p>For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.</p>
	15:14	<p>Reserved</p> <p>Format: MBZ</p>
	13:0	<p>Source X offset Format U14. This field specifies the horizontal offset in pixels from Surface Base Address to the start (origin) of the surface.</p> <p style="text-align: center;">Programming Notes</p> <p>For Linear surface X offset must be 0. For Tiled surface X offset must be greater than or equal to 0 and less than 16K in pixels.</p>
15	31:16	<p>Reserved</p> <p>Format: MBZ</p>
	15:0	<p>Destination Clear Address High</p> <p>Format: GraphicsAddress[47:32]</p> <p>Specifies the higher bits of Graphics Address for destination surface where clear value is stored in the from of RGBA (R in the LSB and A in the MSB - in that order).</p>

XY_COLOR_BLT

XY_COLOR_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>COLOR_BLT is the simplest BLT operation. It performs a color fill to the destination (with a possible ROP). The only operand is the destination operand which is written dependent on the raster operation. The solid pattern color is stored in the pattern background register.</p> <p>This instruction is optimized to run at the maximum memory write bandwidth.</p> <p>The typical (and fastest) Raster operation code = F0 which performs a copy of the pattern background register to the destination.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	50h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		1xb	Write Alpha Channel
x1b	Write RGB Channel		
19:12	Reserved		
	Format:	MBZ	
11	Tiling Enable		
	Value	Name	
	0b	Tiling Disabled (Linear Blit)	
	1b	Tiling Enabled	
10:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	05h	
	Format:	=n	
1 BR13	31	Reserved	
	Format:	MBZ	
	30	Clipping Enabled	
		Value	Name

XY_COLOR_BLT												
		<table border="1"> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </table>	0b	Disabled	1b	Enabled						
0b	Disabled											
1b	Enabled											
	29:26	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	25:24	Color Depth <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name											
00b	8 Bit Color											
01b	16 Bit Color(565)											
10b	16 Bit Color(1555)											
11b	32 Bit Color											
	23:16	Raster Operation 15:0 Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.										
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.										
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.										
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.										
4.5	63:0	Destination Base Address <table border="1"> <tr> <td>Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]											
6 BR16	31:0	Solid Pattern Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]										

XY_FAST_COLOR_BLT

XY_FAST_COLOR_BLT			
Source:	BlitterCS		
Length Bias:	2		
Description			
<p>XY_FAST_COLOR_BLT instruction performs a color blit where the only operands involved are an in-line color source and destination surface of the same bit width. This new blit command will happen in large numbers, consecutively, possibly an entire batch will comprise only new blit commands Legacy commands and new blit command will not be interspersed. If they are, they will be separated by implied HW flush: Whenever there is a transition between this new Fast Blit commands and the Legacy Blit commands (2D BLT instructions other than XY_FAST_COPY_BLT, XY_BLOCK_COPY_BLT, XY_FAST_COLOR_BLT), the HW will impose an automatic flush BEFORE the execution (at the beginning) of the next blitter command. The starting pixel of Fast Colorblit for destination should be on pixel boundary.</p> <p>Note that when two sequential fast copy blits have different source surfaces, but their destinations refer to the same destination surfaces and therefore destinations overlap it is imperative that a Flush be inserted between the two blits.</p>			
DWord	Bit	Description	
0	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Format:	Opcode
		Value	Name
		44h	XY_FAST_COLOR_BLT [Default]
	21:19	Color Depth	
		Description	
		This field actually programs bits per pixel value for each pixel of the surface. Reprogramming these bits require explicit flush of Copy Engine.	
Value		Name	
000b		8 bit color [Default]	
001b		16 bit color	
010b		32 bit color	
011b		64 bit color	
100b		96 bit color (only supported for linear case)	
101b		128 bit color	
110b	RESERVED		

XY_FAST_COLOR_BLT								
	111b	RESERVED						
	18:14	Reserved Format: _____ MBZ						
	13:9	Reserved Format: _____ MBZ						
	8	Reserved Format: _____ MBZ						
	7:0	DWord Length Format: _____ =n Total Length - 2 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td>09h</td> <td>Excludes DWORD 0,1 [Default]</td> </tr> </tbody> </table>	Value	Name	09h	Excludes DWORD 0,1 [Default]		
Value	Name							
09h	Excludes DWORD 0,1 [Default]							
1	31:30	Destination Tiling These bits indicate destination tiling method. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 30%;">Name</th> <th style="width: 40%;">Description</th> </tr> </thead> <tbody> <tr> <td>01b</td> <td>YMAJOR</td> <td>Y major tiling</td> </tr> </tbody> </table>	Value	Name	Description	01b	YMAJOR	Y major tiling
	Value	Name	Description					
	01b	YMAJOR	Y major tiling					
	29:28	Reserved Format: _____ MBZ						
	27:21	Destination MOCS value MOCS (Memory Object State Control) for destination operand.						
20:18	Reserved Format: _____ MBZ							
17:0	Destination Pitch Format: _____ U18-1 For Linear Surfaces , the pitch must be multiple of pixel width in bytes. For Tiled surfaces, the pitch has to be a multiple of the Tile Width (X direction width of the Tile) expressed in dwords (4 bytes).							
2	31:16	Destination Y1 Coordinate (Top) The destination start line (inclusive) for Fast Color blit. Format is 16-bit signed number.						
	15:0	Destination X1 Coordinate (Left) The destination start pixel (inclusive) for Fast Color blit. Format is 16-bit signed number.						
3	31:16	Destination Y2 Coordinate (Bottom) The destination end line (exclusive) for Fast Color blit. Format is 16-bit signed number.						
	15:0	Destination X2 Coordinate (Right) The destination end pixel (exclusive) for Fast Color blit.						

XY_FAST_COLOR_BLT			
		Format is 16-bit signed number.	
4..5	63:0	Destination Base Address	
		Format: GraphicsAddress[63:0]	
Description			
This bitfield contains the base address of the destination surface. When Tiling is enabled this address is cacheline (64Byte) aligned. When Destination Tiling is disabled, this address is byte aligned.			
6	31	Reserved	
	Format: MBZ		
	30	Reserved	
	Format: MBZ		
	29:16	Destination Y offset	
Description			
Format is U14. This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface.			
Programming Notes			
For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.			
15:14	Reserved		
Format: MBZ			
7..10	127:0	Fill Color	
		Description	
		The Dwords contains Color data to use for the fill operation. Format depends on the color depth selected and is always packed little endian way starting with DW[7][0]. 8 bit Color Format: DW[7][7:0]	

XY_FAST_COLOR_BLT				
		16 bit Color Format: DW[7][15:0] 32 bit Color Format: DW[7][31:0] 64 bit Color Format: DW[8], DW[7] 96 bit Color Format: DW[9], DW[8], DW[7] 128 bit Color Format: DW[10], DW[9], DW[8], DW[7]		
12	31:16	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	Destination Clear Address High <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td style="width: 70%;">GraphicsAddress[47:32]</td> </tr> </table> <p>Specifies the higher bits of Graphics Address of the destination surface where clear value is stored in the form of RGBA (R in the LSB and A in the MSB - in that order)</p>	Format:	GraphicsAddress[47:32]	
Format:	GraphicsAddress[47:32]			

XY_FAST_COPY_BLT

XY_FAST_COPY_BLT				
Source:	BlitterCS			
Length Bias:	2			
Description				
<p>This BLT instruction performs a color source copy where the only operands involved are a color source and destination of the same bit width. Note that this command does not support Clipping operations. This new blit command will happen in large numbers, consecutively, possibly an entire batch will comprise only new blit commands Legacy commands and new blit command will not be interspersed. If they are, they will be separated by implied HW flush: Whenever there is a transition between this new Fast Blit command and the Legacy Blit commands (2D BLT instructions other than XY_BLOCK_COPY_BLT, XY_FAST_COPY_BLT and XY_FAST_COLOR_BLT), the HW will impose an automatic flush BEFORE the execution (at the beginning) of the next blitter command. New blit command can use any combination of memory surface type - linear, tiledX, tiledY, and the tiling information is conveyed as part of the new Fast Copy command. The Fast Copy Blit supports the new 64KB Tiling. The starting pixel of Fast Copy blit for both source and destination should be on an OWord boundary.</p> <p>Note that when two sequential fast copy blits have different source surfaces, but their destinations refer to the same destination surfaces and therefore destinations overlap it is imperative that a Flush be inserted between the two blits.</p>				
DWord	Bit	Description		
0 BR00	31:29	Client		
		Default Value:	02h 2D Processor	
		Format:	Opcode	
	28:22	Instruction Target(Opcode)		
		Default Value:	42h	
		Format:	Opcode	
	21:20	Source Tiling Method		
		SW is required to flush the HW before changing the polarity of these bits for subsequent blits.		
		Value	Name	Description
		00b	Linear (Tiling Disabled)	
10b		YMAJOR	Choosing between 'Legacy Tile-Y' or the 'Tile4' can be done in DWord 1, Bit[31].	
11b	Tile64			
19:15	Reserved			
	Format:	MBZ		
14:13	Destination Tiling Method			
	SW is required to flush the HW before changing the polarity of these bits for subsequent blits.			

XY_FAST_COPY_BLT

Value	Name	Description																				
00b	Linear (Tiling Disabled)																					
10b	YMAJOR	Choosing between 'Legacy Tile-Y' or the 'Tile4' can be done in DWord 1, Bit[30].																				
11b	Tile64																					
12:8	Reserved																					
	Format:	MBZ																				
7:0	DWord Length																					
	Default Value:	08h Excludes DWORD 0,1																				
	Format:	=n																				
	08h																					
1 BR13	31	Tile Y Type for Source Source being Tile-Y can be selected in DWord 0, Bit[21:20].																				
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="background-color: #e1eef6;">Value</th> <th style="background-color: #e1eef6;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Legacy Tile-Y</td> </tr> <tr> <td>1b</td> <td>Tile4</td> </tr> </tbody> </table>	Value	Name	0b	Legacy Tile-Y	1b	Tile4														
	Value	Name																				
	0b	Legacy Tile-Y																				
	1b	Tile4																				
		0b	Legacy Tile-Y																			
		1b	Tile4																			
	30	Tile Y Type for Destination Destination being Tile-Y can be selected in DWord 0, Bit[14:13].																				
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="background-color: #e1eef6;">Value</th> <th style="background-color: #e1eef6;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Legacy Tile-Y</td> </tr> <tr> <td>1b</td> <td>Tile4</td> </tr> </tbody> </table>	Value	Name	0b	Legacy Tile-Y	1b	Tile4														
	Value	Name																				
0b	Legacy Tile-Y																					
1b	Tile4																					
	0b	Legacy Tile-Y																				
	1b	Tile4																				
29:28	Reserved																					
	Format:	MBZ																				
27	Reserved																					
	Format:	MBZ																				
26:24	Color Depth																					
	<table border="1" style="width: 100%;"> <thead> <tr> <th style="background-color: #e1eef6;">Value</th> <th style="background-color: #e1eef6;">Name</th> <th style="background-color: #e1eef6;">Programming Notes</th> </tr> </thead> <tbody> <tr> <td>000b</td> <td>8 bit color</td> <td></td> </tr> <tr> <td>001b</td> <td>16 bit color (565)</td> <td></td> </tr> <tr> <td>010b</td> <td>RESERVED</td> <td>[] Programming of 010b is not supported.</td> </tr> <tr> <td>011b</td> <td>32 bit color</td> <td></td> </tr> <tr> <td>100b</td> <td>64 bit color (for 64KB Tiling)</td> <td></td> </tr> <tr> <td>101b</td> <td>128 bit color (for 64KB Tiling)</td> <td></td> </tr> </tbody> </table>	Value	Name	Programming Notes	000b	8 bit color		001b	16 bit color (565)		010b	RESERVED	[] Programming of 010b is not supported.	011b	32 bit color		100b	64 bit color (for 64KB Tiling)		101b	128 bit color (for 64KB Tiling)	
Value	Name	Programming Notes																				
000b	8 bit color																					
001b	16 bit color (565)																					
010b	RESERVED	[] Programming of 010b is not supported.																				
011b	32 bit color																					
100b	64 bit color (for 64KB Tiling)																					
101b	128 bit color (for 64KB Tiling)																					
	000b	8 bit color																				
	001b	16 bit color (565)																				
	010b	RESERVED [] Programming of 010b is not supported.																				
	011b	32 bit color																				
	100b	64 bit color (for 64KB Tiling)																				
	101b	128 bit color (for 64KB Tiling)																				
23:16	Reserved																					

XY_FAST_COPY_BLT				
	15:0	Destination Pitch <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table>	Format:	U16
Format:	U16			
2 BR22	31:16	Destination Y1 Coordinate (Top) <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S15</td> </tr> </table> Destination start line (inclusive) for Fast Copy blit.	Format:	S15
	Format:	S15		
15:0	Destination X1 Coordinate (Left) <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S15</td> </tr> </table> Destination start pixel (inclusive) for Fast Copy blit. It should be on an OWord boundary.	Format:	S15	
Format:	S15			
3 BR23	31:16	Destination Y2 Coordinate (Bottom) <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S15</td> </tr> </table> Destination end line (exclusive) for Fast Copy blit.	Format:	S15
	Format:	S15		
15:0	Destination X2 Coordinate (Right) <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S15</td> </tr> </table> Destination end pixel (exclusive) for Fast Copy blit.	Format:	S15	
Format:	S15			
4..5	63:0	Destination Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Format:</td> <td style="width: 60%;">VIRTUAL_ADDR[63:0]</td> </tr> </table> This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
6 BR26	31:16	Source Y1 Coordinate (Top) <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">S15</td> </tr> </table> Source start line (inclusive) for Fast Copy blit.	Format:	S15
	Format:	S15		
15:0	Source X1 Coordinate (Left) Source start pixel (inclusive) for Fast Copy blit. It should be on an OWord boundary.			
7 BR11	31:16	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	Source Pitch <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td style="width: 30%;">U16</td> </tr> </table>	Format:	U16	
Format:	U16			
8..9	63:0	Source Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 40%;">Format:</td> <td style="width: 60%;">VIRTUAL_ADDR[63:0]</td> </tr> </table> This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			

XY_FULL_BLT

XY_FULL_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source and pattern operands are the same bit width as the destination operand.</p> <p>The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	55h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
x1b	Write RGB Channel		
19:16	Reserved		
	Format:	MBZ	
15	Src Tiling Enable		
	Value	Name	Description
	0b	Tiling Disabled (Linear Blit)	
	1b	Tiling Enabled	Tile-X or Tile-Y.
14:12	Pattern Horizontal Seed		

XY_FULL_BLT											
		Pixel of the scan line to start on corresponding to DST X=0.									
	11	Dest Tiling Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td>1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
Value	Name	Description									
0b	Tiling Disabled (Linear Blit)										
1b	Tiling Enabled	Tile-X or Tile-Y.									
	10:8	Pattern Vertical Seed Starting scan line of the 8x8 pattern corresponding to DST Y=0.									
	7:0	DWord Length Default Value: <table border="1" style="display: inline-table;"><tr><td>0Ah</td></tr></table>	0Ah								
0Ah											
1 BR13	31	Reserved Format: <table border="1" style="display: inline-table;"><tr><td>MBZ</td></tr></table>	MBZ								
	MBZ										
	30	Clipping Enabled <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled			
	Value	Name									
	0b	Disabled									
	1b	Enabled									
	29:26	Reserved Format: <table border="1" style="display: inline-table;"><tr><td>MBZ</td></tr></table>	MBZ								
MBZ											
25:24	Color Depth <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name										
00b	8 Bit Color										
01b	16 Bit Color(565)										
10b	16 Bit Color(1555)										
11b	32 Bit Color										
23:16	Raster Operation										
15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KWords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.									
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.									
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.									
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.									
4.5	63:0	Destination Base Address Format: <table border="1" style="display: inline-table;"><tr><td>VIRTUAL_ADDR[63:0]</td></tr></table> Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this	VIRTUAL_ADDR[63:0]								
VIRTUAL_ADDR[63:0]											

XY_FULL_BLT				
		address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.		
6 BR11	31:16	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td>MBZ</td></tr></table>		MBZ
		MBZ		
15:0	Source Pitch (double word aligned and signed) and in DWords 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).			
7 BR26	31:16	Source Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Source X1 Coordinate (Left) 16 bit signed number.		
8..9	63:0	Source Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td>VIRTUAL_ADDR[63:0]</td></tr></table> Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.		VIRTUAL_ADDR[63:0]
	VIRTUAL_ADDR[63:0]			
10..11	63:0	Pattern Base Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 150px;"></td><td>VIRTUAL_ADDR[63:0]</td></tr></table> Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.		VIRTUAL_ADDR[63:0]
	VIRTUAL_ADDR[63:0]			

XY_FULL_IMMEDIATE_PATTERN_BLT

XY_FULL_IMMEDIATE_PATTERN_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source and immediate pattern operands are the same bit width as the destination operand. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64 DWs) for 8, 16, and 32 bpp color patterns. DWL indicates the total number of Dwords of immediate data.</p> <p>The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	74h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
x1b	Write RGB Channel		
19:16	Reserved		
	Format:	MBZ	
15	Src Tiling Enable		
	Value	Name	Description
	0b	Tiling Disabled (Linear)	
	1b	Tiling Enabled	Tile-X or Tile-Y.

XY_FULL_IMMEDIATE_PATTERN_BLT

	14:12	Pattern Horizontal Seed (pixel of the scan line to start on corresponding to DST X=0)	
	11	Dest Tiling Enable	
		Value	Name
		0b	Tiling Disabled (Linear Blit)
		1b	Tiling Enabled
			Tile-X or Tile-Y.
	10:8	Pattern Vertical Seed Starting scan line of the 8x8 pattern corresponding to DST Y=0.	
	7:0	DWord Length	
		Format:	=n
		n = 08 + DWL = (where 'DWL' is Number of Immediate data in terms of double words). Immediate data size is 16 DW for 8 BPP, 32 DW for 16 BPP and 64 DW for 32 BPP.	
		Value	Name
		[24,72]	Excludes DWORD 0,1
1 BR13	31	Reserved	
		Format:	MBZ
	30	Clipping Enabled	
		Value	Name
		0b	Disabled
		1b	Enabled
	29:26	Reserved	
		Format:	MBZ
	25:24	Color Depth	
		Value	Name
		00b	8 Bit Color
		01b	16 Bit Color(565)
		10b	16 Bit Color(1555)
		11b	32 Bit Color
	23:16	Raster Operation	
	15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).	
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.	
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.	
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.	

XY_FULL_IMMEDIATE_PATTERN_BLT				
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.		
4..5	63:0	Destination Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> Base address of the destination surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
6 BR11	31:16	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	Source Pitch (double word aligned and signed) and in DWords 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).			
7 BR26	31:16	Source Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Source X1 Coordinate (Left) 16 bit signed number.		
8..9	63:0	Source Address <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
10..n	31:0	Immediate Data 0		

XY_FULL_MONO_PATTERN_BLT

XY_FULL_MONO_PATTERN_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The pattern operand is monochrome and the source operand is the same bit width as the destination operand.</p> <p>The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the pattern foreground color is used in the ROP operation.</p> <p>All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8. Setting both Solid Pattern Select = 1 and Mono Pattern Transparency = 1 is mutually exclusive. The device implementation results in NO PIXELS DRAWN.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	57h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
x1b	Write RGB Channel		
19:16	Reserved		
	Format:	MBZ	
15	Src Tiling Enable		

XY_FULL_MONO_PATTERN_BLT												
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td>1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.	
	Value	Name	Description									
	0b	Tiling Disabled (Linear Blit)										
	1b	Tiling Enabled	Tile-X or Tile-Y.									
	14:12	Pattern Horizontal Seed (pixel of the scan line to start on corresponding to DST X=0)										
	11	Dest Tiling Enable <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td>1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>		Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
	Value	Name	Description									
	0b	Tiling Disabled (Linear Blit)										
	1b	Tiling Enabled	Tile-X or Tile-Y.									
	10:8	Pattern Vectical Seed Starting scan line of the 8x8 pattern corresponding to DST Y=0.										
7:0	DWord Length <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0Ch</td> <td></td> </tr> </tbody> </table>		Value	Name	0Ch							
Value	Name											
0Ch												
1 BR13	31	Solid Pattern Select <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Solid Pattern</td> </tr> <tr> <td>1</td> <td>Solid Pattern</td> </tr> </tbody> </table>		Value	Name	0	No Solid Pattern	1	Solid Pattern			
	Value	Name										
	0	No Solid Pattern										
	1	Solid Pattern										
	30	Clipping Enabled <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>		Value	Name	0b	Disabled	1b	Enabled			
	Value	Name										
	0b	Disabled										
	1b	Enabled										
29	Reserved Format: _____ MBZ											
28:27	Mono Source Transparency Mode <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Use Background</td> </tr> <tr> <td>1</td> <td>Transparency Enabled</td> </tr> </tbody> </table>		Value	Name	0	Use Background	1	Transparency Enabled				
Value	Name											
0	Use Background											
1	Transparency Enabled											
26	Reserved Format: _____ MBZ											
25:24	Color Depth <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>		Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name											
00b	8 Bit Color											
01b	16 Bit Color(565)											
10b	16 Bit Color(1555)											
11b	32 Bit Color											

XY_FULL_MONO_PATTERN_BLT				
	23:16	Raster Operation		
	15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).		
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.		
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.		
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.		
4..5	63:0	Destination Base Address <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> <p>Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.</p>	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
6 BR11	31:16	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ
	Format:	MBZ		
15:0	Source Pitch (double word aligned and signed) and in DWords 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).			
7 BR26	31:16	Source Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Source X1 Coordinate (Left) 16 bit signed number.		
8..9	63:0	Source Address <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> <p>Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.</p>	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
10 BR16	31:0	Pattern Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		
11 BR17	31:0	Pattern Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		
12 BR20	31:0	Pattern Data 0 (least significant DW)		
13 BR21	31:0	Pattern Data 1 (most significant DW)		

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

XY_FULL_MONO_PATTERN_MONO_SRC_BLT									
Source:	BlitterCS								
Length Bias:	2								
<p>The full BLT provides the ability to specify all 3 operands: destination, source, and pattern. The pattern and source operands are monochrome.</p> <p>The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation.</p> <p>All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the destination X1 coordinate.</p> <p>The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the source bit is 1, then the pattern foreground color is used in the ROP operation. The monochrome source transparency mode works identical to the pattern transparency mode.</p> <p>All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p> <p>Setting both Solid Pattern Select = 1 and Mono Pattern Transparency = 1 is mutually exclusive. The device implementation results in NO PIXELS DRAWN.</p> <p>Negative Stride (= Pitch) is NOT ALLOWED.</p>									
DWord	Bit	Description							
0 BR00	31:29	Client <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode			
	Default Value:	02h 2D Processor							
	Format:	Opcode							
	28:22	Instruction Target(Opcode) <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>58h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	58h	Format:	Opcode			
	Default Value:	58h							
Format:	Opcode								
21:20	32bpp Byte Mask This field is only used for 32bpp. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">1xb</td> <td style="text-align: center;">Write Alpha Channel</td> </tr> <tr> <td style="text-align: center;">x1b</td> <td style="text-align: center;">Write RGB Channel</td> </tr> </tbody> </table>	Value	Name	00b	[Default]	1xb	Write Alpha Channel	x1b	Write RGB Channel
Value	Name								
00b	[Default]								
1xb	Write Alpha Channel								
x1b	Write RGB Channel								
19:17	Monochrome source data bit position of the first pixel within a byte per scan line.								
16:15	Reserved								

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

		Format:	MBZ									
	14:12	Pattern Horizontal Seed (pixel of the scan line to start on corresponding to DST X=0)										
	11	Tiling Enable <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 55%;">Name</th> <th style="width: 30%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>		Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
Value	Name	Description										
0b	Tiling Disabled (Linear Blit)											
1b	Tiling Enabled	Tile-X or Tile-Y.										
	10:8	Pattern Vertical Seed Starting scan line of the 8x8 pattern corresponding to DST Y = 0.										
	7:0	DWord Length <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0Ch</td> <td></td> </tr> </tbody> </table>		Value	Name	0Ch						
Value	Name											
0Ch												
1 BR13	31	Solid Pattern Select <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>No Solid Pattern</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Solid Pattern</td> </tr> </tbody> </table>		Value	Name	0	No Solid Pattern	1	Solid Pattern			
Value	Name											
0	No Solid Pattern											
1	Solid Pattern											
	30	Clipping Enabled <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 70%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Disabled</td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Enabled</td> </tr> </tbody> </table>		Value	Name	0b	Disabled	1b	Enabled			
Value	Name											
0b	Disabled											
1b	Enabled											
	29	Mono Source Transparency Mode <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Use Background</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Transparency Enabled</td> </tr> </tbody> </table>		Value	Name	0	Use Background	1	Transparency Enabled			
Value	Name											
0	Use Background											
1	Transparency Enabled											
	28	Mono Pattern Transparency Mode <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Use Background</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Transparency Enabled</td> </tr> </tbody> </table>		Value	Name	0	Use Background	1	Transparency Enabled			
Value	Name											
0	Use Background											
1	Transparency Enabled											
	27:26	Reserved Format: MBZ										
	25:24	Color Depth <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 20%;">Value</th> <th style="width: 80%;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td>8 Bit Color</td> </tr> <tr> <td style="text-align: center;">01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td style="text-align: center;">10b</td> <td>16 Bit Color(1555)</td> </tr> </tbody> </table>		Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	
Value	Name											
00b	8 Bit Color											
01b	16 Bit Color(565)											
10b	16 Bit Color(1555)											

XY_FULL_MONO_PATTERN_MONO_SRC_BLT				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">11b</td> <td>32 Bit Color</td> </tr> </table>	11b	32 Bit Color
11b	32 Bit Color			
	23:16	Raster Operation		
	15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).		
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.		
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.		
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.		
4..5 This bitfield contains the base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	Destination Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	Mono Source Address <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
8 BR18	31:0	Source Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		
9 BR19	31:0	Source Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		
10 BR16	31:0	Pattern Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		
11 BR17	31:0	Pattern Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		
12 BR20	31:0	Pattern Data 0 (least significant DW)		
13 BR21	31:0	Pattern Data 1 (most significant DW)		

XY_FULL_MONO_SRC_BLT

XY_FULL_MONO_SRC_BLT									
Source:	BlitterCS								
Length Bias:	2								
<p>The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source operand is monochrome and the pattern operand is the same bit width as the destination.</p> <p>The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation.</p> <p>All non-text and non-immediate monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the Destination X1 coordinate.</p> <p>All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p> <p>Negative Stride (= Pitch) is NOT ALLOWED</p>									
DWord	Bit	Description							
0 BR00	31:29	Client <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode			
	Default Value:	02h 2D Processor							
	Format:	Opcode							
	28:22	Instruction Target(Opcode) <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>56h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	56h	Format:	Opcode			
	Default Value:	56h							
	Format:	Opcode							
	21:20	32bpp Byte Mask This field is only used for 32bpp. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">1xb</td> <td>Write Alpha Channel</td> </tr> <tr> <td style="text-align: center;">x1b</td> <td>Write RGB Channel</td> </tr> </tbody> </table>	Value	Name	00b	[Default]	1xb	Write Alpha Channel	x1b
Value	Name								
00b	[Default]								
1xb	Write Alpha Channel								
x1b	Write RGB Channel								
19:17	Monochrome source data bit position of the first pixel within a byte per scan line.								
16:15	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ								
14:12	Pattern Horizontal Seed (pixel of the scan line to start on corresponding to DST X=0)								

XY_FULL_MONO_SRC_BLT

	11	Tiling Enable	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td>1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
	Value	Name	Description									
	0b	Tiling Disabled (Linear Blit)										
	1b	Tiling Enabled	Tile-X or Tile-Y.									
10:8	Pattern Vertical Seed	Starting scan line of the 8x8 pattern corresponding to DST Y = 0.										
7:0	DWord Length	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0Ah</td> <td></td> </tr> </tbody> </table>	Value	Name	0Ah							
Value	Name											
0Ah												
1 BR13	31	Reserved	Format: MBZ									
	30	Clipping Enabled	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled			
	Value	Name										
	0b	Disabled										
	1b	Enabled										
	29	Mono Source Transparency Mode	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Use Background</td> </tr> <tr> <td>1</td> <td>Transparency Enabled</td> </tr> </tbody> </table>	Value	Name	0	Use Background	1	Transparency Enabled			
	Value	Name										
0	Use Background											
1	Transparency Enabled											
28:26	Reserved	Format: MBZ										
25:24	Color Depth	<table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name											
00b	8 Bit Color											
01b	16 Bit Color(565)											
10b	16 Bit Color(1555)											
11b	32 Bit Color											
23:16	Raster Operation											
15:0	Destination Pitch in DWords	2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top)	16 bit signed number.									
	15:0	Destination X1 Coordinate (Left)	16 bit signed number.									
3	31:16	Destination Y2 Coordinate (Bottom)										

XY_FULL_MONO_SRC_BLT			
BR23		16 bit signed number.	
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.	
4.5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.	63:0	Destination Base Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;">VIRTUAL_ADDR[63:0]</td></tr></table>	VIRTUAL_ADDR[63:0]
VIRTUAL_ADDR[63:0]			
6.7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	Mono Source Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;">VIRTUAL_ADDR[63:0]</td></tr></table>	VIRTUAL_ADDR[63:0]
VIRTUAL_ADDR[63:0]			
8 BR18	31:0	Source Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]	
9 BR19	31:0	Source Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]	
10..11 28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The programmed Pattern Base Address must always be Cache Line (64byte) aligned.	63:0	Pattern Base Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;">VIRTUAL_ADDR[63:0]</td></tr></table>	VIRTUAL_ADDR[63:0]
VIRTUAL_ADDR[63:0]			

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source operand is a monochrome and the immediate pattern operand is the same bit width as the destination. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns. The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the destination X1 coordinate. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation. The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8. Negative Stride (= Pitch) is NOT ALLOWED.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	75h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
x1b	Write RGB Channel		
19:17	Monochrome source data bit position of the first pixel within a byte per scan line.		
16:15	Reserved		
	Format:	MBZ	
14:12	Pattern Horizontal Seed (pixel of the scan line to start on corresponding to DST X=0)		
11	Tiling Enable		

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

		Value	Name	Description									
		0b	Tiling Disabled (Linear Blit)										
		1b	Tiling Enabled	Tile-X or Tile-Y.									
		10:8	Pattern Vertical Seed Starting scan line of the 8x8 pattern corresponding to DST Y=0.										
		7:0	DWord Length Format: _____ =n n = 08 + DWL , (where 'DWL' is Number of Immediate data in terms of double words). Immediate data size is 16 DW for 8 BPP, 32 DW for 16 BPP and 64 DW for 32 BPP.										
			Value	Name									
			[24,72]	Excludes DWORD 0,1									
1 BR13	31	Reserved Format: _____ MBZ											
	30	Clipping Enabled <table border="1" style="width: 100%;"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>			Value	Name	0b	Disabled	1b	Enabled			
	Value	Name											
	0b	Disabled											
	1b	Enabled											
	29	Mono Source Transparency Mode <table border="1" style="width: 100%;"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Use Background</td> </tr> <tr> <td>1</td> <td>Transparency Enabled</td> </tr> </tbody> </table>			Value	Name	0	Use Background	1	Transparency Enabled			
	Value	Name											
	0	Use Background											
	1	Transparency Enabled											
	28:26	Reserved Format: _____ MBZ											
25:24	Color Depth <table border="1" style="width: 100%;"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>			Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name												
00b	8 Bit Color												
01b	16 Bit Color(565)												
10b	16 Bit Color(1555)												
11b	32 Bit Color												
23:16	Raster Operation												
15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).												
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.											

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT		
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	Destination Base Address Format: VIRTUAL_ADDR[63:0]
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	Mono Source Address Format: VIRTUAL_ADDR[63:0]
8 BR18	31:0	Source Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	Source Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10..n	31:0	Immediate Data

XY_MONO_PAT_BLT

XY_MONO_PAT_BLT										
Source:	BlitterCS									
Length Bias:	2									
<p>MONO_PAT_BLT is used when we have no source and the monochrome pattern is not trivial (is not a solid color only). The monochrome pattern is loaded from the instruction stream.</p> <p>All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p> <p>The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the pattern bit is 1, then the pattern foreground color is used in the ROP operation.</p>										
DWord	Bit	Description								
0 BR00	31:29	Client <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode				
	Default Value:	02h 2D Processor								
	Format:	Opcode								
	28:22	Instruction Target(Opcode) <table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>52h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	52h	Format:	Opcode				
	Default Value:	52h								
	Format:	Opcode								
	21:20	32bpp Byte Mask This field is only used for 32bpp. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">1xb</td> <td>Write Alpha Channel</td> </tr> <tr> <td style="text-align: center;">x1b</td> <td>Write RGB Channel</td> </tr> </tbody> </table>	Value	Name	00b	[Default]	1xb	Write Alpha Channel	x1b	Write RGB Channel
	Value	Name								
	00b	[Default]								
	1xb	Write Alpha Channel								
x1b	Write RGB Channel									
19:15	Reserved <table border="1" style="width: 100%; margin-top: 5px;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
14:12	Pattern Horizontal Seed Pixel of the scan line to start on corresponding to DST X=0.									
11	Tiling Enable <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
Value	Name	Description								
0b	Tiling Disabled (Linear Blit)									
1b	Tiling Enabled	Tile-X or Tile-Y.								
10:8	Pattern Vertical Seed Scan line of the 8x8 pattern to start on corresponding to DST Y=0.									

XY_MONO_PAT_BLT

	7:0	DWord Length										
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>08h</td> <td></td> </tr> </tbody> </table>	Value	Name	08h							
Value	Name											
08h												
1 BR13	31	Reserved										
		Format: MBZ										
	30	Clipping Enabled										
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled				
	Value	Name										
	0b	Disabled										
	1b	Enabled										
29	Reserved											
	Format: MBZ											
28	Mono Pattern Transparency Mode											
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Use Background</td> </tr> <tr> <td>1</td> <td>Transparency Enabled</td> </tr> </tbody> </table>	Value	Name	0	Use Background	1	Transparency Enabled					
Value	Name											
0	Use Background											
1	Transparency Enabled											
	27:26	Reserved										
		Format: MBZ										
	25:24	Color Depth										
		<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name											
00b	8 Bit Color											
01b	16 Bit Color(565)											
10b	16 Bit Color(1555)											
11b	32 Bit Color											
	23:16	Raster Operation										
	15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.										
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.										
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.										
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.										
4.5 Base address of the destination surface: X=0,	63:0	Destination Base Address										
		Format: VIRTUAL_ADDR[63:0]										

XY_MONO_PAT_BLT		
Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.		
6 BR16	31:0	Pattern Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR17	31:0	Pattern Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
8 BR20	31:0	Pattern Data 0
9 BR21	31:0	Pattern Data 1

XY_MONO_PAT_FIXED_BLT

XY_MONO_PAT_FIXED_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>MONO_PAT_FIXED_BLT is used when we have no source and the monochrome pattern is not trivial (is not a solid color only). The monochrome pattern is one of 10 fixed patterns described below. The pattern seeds can still be used with the fixed patterns, creating even more fixed patterns. This eliminates 2 doublewords compared to the XY_MONO_PAT_BLT command packet.</p> <p>All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p> <p>The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the pattern bit is 1, then the pattern foreground color is used in the ROP operation.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	59h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
x1b	Write RGB Channel		
19	Reserved		
	Format:	MBZ	
18:15	Fixed Pattern		
	Value	Name	
	0000b	HS_HORIZONTAL	
	0001b	HS_VERTICAL	
	0010b	HS_FDIAGONAL	
	0011b	HS_BDIAGONAL	
	0100b	HS_CROSS	

XY_MONO_PAT_FIXED_BLT

		0101b	HS_DIAGCROSS
		0110b	Reserved
		0111b	Reserved
		1000b	Screen Door
		1001b	SD Wide
		1010b	Walking Bit (one)
		1011b	Walking Zero
		1100b	Reserved
		1101b	Reserved
		1110b	Reserved
		1111b	Reserved
	14:12	Pattern Horizontal Seed Pixel of the scan line to start on corresponding to DST X=0.	
	11	Tiling Enable	
		Value	Name
		0b	Tiling Disabled (Linear Blit)
		1b	Tiling Enabled
			Tile-X or Tile-Y.
	10:8	Pattern Vertical Seed Scan line of the 8x8 pattern to start on corresponding to DST Y=0.	
	7:0	DWord Length	
		Format:	=n
		Value	Name
		06h	
1 BR13	31	Reserved	
		Format:	MBZ
	30	Clipping Enabled	
		Value	Name
		0b	Disabled
	1b	Enabled	
	29	Reserved	
		Format:	MBZ
	28	Mono Pattern Transparency Mode	
		Value	Name
		0	Use Background
		1	Transparency Enabled
	27:26	Reserved	

XY_MONO_PAT_FIXED_BLT												
		<table border="1" style="width: 100%;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	25:24	Color Depth <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 30%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name											
00b	8 Bit Color											
01b	16 Bit Color(565)											
10b	16 Bit Color(1555)											
11b	32 Bit Color											
	23:16	Raster Operation										
	15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.										
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.										
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.										
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.										
4..5	63:0	Destination Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]											
6 BR16	31:0	Pattern Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]										
7 BR17	31:0	Pattern Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]										

XY_MONO_SRC_COPY_BLT

XY_MONO_SRC_COPY_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>This BLT instruction performs a monochrome source copy where the only operands involved is a monochrome source and destination. The source and destination operands cannot overlap therefore the X and Y directions are always forward.</p> <p>All non-text monochrome sources are word aligned. At the end of a scan line of monochrome source, all bits until the next word boundary are ignored. The monochrome source data bit position field [2:0] indicates the bit position within the first byte of the scan line that should be used as the first source pixel which corresponds to the destination X1 coordinate.</p> <p>The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen must involve source and no pattern data in the ROP operation. Negative Stride (= Pitch) is NOT ALLOWED.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value: 02h 2D Processor	
	Format: Opcode		
	28:22	Instruction Target(Opcode)	
		Default Value: 54h	
	Format: Opcode		
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb	Write Alpha Channel		
x1b	Write RGB Channel		
19:17	Monochrome source data bit position of the first pixel within a byte per scan line.		
16:12	Reserved		
	Format: MBZ		
11	Tiling Enable		
	Value	Name	Description
	0b	Tiling Disabled (Linear Blit)	
	1b	Tiling Enabled	Tile-X or Tile-Y.
10:8	Reserved		
	Format: MBZ		

XY_MONO_SRC_COPY_BLT											
	7:0	DWord Length <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">08h</td> <td></td> </tr> </tbody> </table>	Value	Name	08h						
Value	Name										
08h											
1 BR13	31	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
	30	Clipping Enabled <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td style="text-align: center;">Disabled</td> </tr> <tr> <td style="text-align: center;">1b</td> <td style="text-align: center;">Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled			
	Value	Name									
	0b	Disabled									
	1b	Enabled									
29	Mono Source Transparency Mode <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%; text-align: center;">Value</th> <th style="width: 80%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Use Background</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Transparency Enabled</td> </tr> </tbody> </table>	Value	Name	0	Use Background	1	Transparency Enabled				
Value	Name										
0	Use Background										
1	Transparency Enabled										
28:26	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td style="text-align: center;">MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
25:24	Color Depth <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%; text-align: center;">Value</th> <th style="width: 80%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td style="text-align: center;">8 Bit Color</td> </tr> <tr> <td style="text-align: center;">01b</td> <td style="text-align: center;">16 Bit Color(565)</td> </tr> <tr> <td style="text-align: center;">10b</td> <td style="text-align: center;">16 Bit Color(1555)</td> </tr> <tr> <td style="text-align: center;">11b</td> <td style="text-align: center;">32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name										
00b	8 Bit Color										
01b	16 Bit Color(565)										
10b	16 Bit Color(1555)										
11b	32 Bit Color										
23:16	Raster Operation <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%; text-align: center;">15:0</td> <td> Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords). </td> </tr> </table>	15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).								
15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.									
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.									
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.									
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.									
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL	63:0	Destination Base Address <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%;">Format:</td> <td style="text-align: center;">VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]							
Format:	VIRTUAL_ADDR[63:0]										

XY_MONO_SRC_COPY_BLT				
(64byte) aligned.				
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	Mono Source Address <table border="1" style="width: 100%;"> <tr> <td style="width: 20%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			
8 BR18	31:0	Source Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		
9 BR19	31:0	Source Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]		

XY_MONO_SRC_COPY_IMMEDIATE_BLT

XY_MONO_SRC_COPY_IMMEDIATE_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>This instruction allows the Driver to send monochrome data through the instruction stream, eliminating the read latency of the source during command execution.</p> <p>The IMMEDIATE_BLT data MUST transfer an even number of doublewords and the exact number of quadwords. DWL indicates the total number of Dwords of immediate data.</p> <p>All non-text monochrome sources are word aligned. At the end of a scan line of monochrome source, all bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates the bit position within the first byte of the scan line that should be used as the first source pixel which corresponds to the destination X1 coordinate.</p> <p>The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen must involve source and no pattern data in the ROP operation. The monochrome source data supplied corresponds to the Destination X1 and Y1 coordinates.</p> <p>Negative Stride (= Pitch) is NOT ALLOWED.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	71h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
x1b	Write RGB Channel		
19:17	Monochrome source data bit position of the first pixel within a byte per scan line.		
16:12	Reserved		
	Format:	MBZ	
11	Tiling Enable		
	Value	Name	Description
	0b	Tiling Disabled (Linear)	
	1b	Tiling Enabled	Tile-X or Tile-Y.
10:8	Reserved		

XY_MONO_SRC_COPY_IMMEDIATE_BLT											
		<table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ										
	7:0	<p>DWord Length</p> <table border="1"> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>n = 06 + DWL Where DWL is number of immediate data in terms of dwords. Immediate data size is 16 DW for 8 BPP, 32 DW for 16 BPP and 64 DW for 32 BPP.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>[22,70]</td> <td>Excludes DWORD 0,1</td> </tr> </tbody> </table>	Format:	=n	Value	Name	[22,70]	Excludes DWORD 0,1			
Format:	=n										
Value	Name										
[22,70]	Excludes DWORD 0,1										
1 BR13	31	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
	30	<p>Clipping Enabled</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled			
	Value	Name									
	0b	Disabled									
	1b	Enabled									
	29	<p>Mono Source Transparency Mode</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Transparency Enabled</td> </tr> <tr> <td>1b</td> <td>Use Background</td> </tr> </tbody> </table>	Value	Name	0b	Transparency Enabled	1b	Use Background			
Value	Name										
0b	Transparency Enabled										
1b	Use Background										
28:26	<p>Reserved</p> <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
25:24	<p>Color Depth</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name										
00b	8 Bit Color										
01b	16 Bit Color(565)										
10b	16 Bit Color(1555)										
11b	32 Bit Color										
23:16	<p>Raster Operation</p>										
	15:0	<p>Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).</p>									
2 BR22	31:16	<p>Destination Y1 Coordinate (Top) 16 bit signed number.</p>									
	15:0	<p>Destination X1 Coordinate (Left) 16 bit signed number.</p>									
3 BR23	31:16	<p>Destination Y2 Coordinate (Bottom) 16 bit signed number.</p>									



XY_MONO_SRC_COPY_IMMEDIATE_BLT

	15:0	Destination X2 Coordinate (Right) 16 bit signed number.						
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	<table border="1" style="width: 100%;"> <tr> <td colspan="2">Destination Base Address</td> </tr> <tr> <td style="width: 20%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> <tr> <td colspan="2" style="height: 40px;"></td> </tr> </table>	Destination Base Address		Format:	VIRTUAL_ADDR[63:0]		
Destination Base Address								
Format:	VIRTUAL_ADDR[63:0]							
6 BR18	31:0	Source Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]						
7 BR19	31:0	Source Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]						
8..n	31:0	Immediate Data						

XY_PAT_BLT_IMMEDIATE

XY_PAT_BLT_IMMEDIATE			
Source:	BlitterCS		
Length Bias:	2		
<p>PAT_BLT_IMMEDIATE is used when there is no source and the color pattern is not trivial (is not a solid color only) and the pattern is pulled through the command stream. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns.</p> <p>DWL indicates the total number of Dwords of immediate data. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	72h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
x1b	Write RGB Channel		
19:15	Reserved		
	Format:	MBZ	
14:12	Pattern Horizontal Seed		
11	Pixel of the scan line to start on corresponding to DST X=0.		
	Tiling Enable		
	Value	Name	
	0b	Tiling Disabled (Linear Blit)	
1b	Tiling Enabled	Tile-X or Tile-Y.	
10:8	Pattern Vertical Seed		
7:0	Scan line of the 8x8 pattern to start on corresponding to DST Y=0.		
	DWord Length		

XY_PAT_BLT_IMMEDIATE														
	<table border="1"> <tr> <td>Default Value:</td> <td>[20,68] Excludes DWORD 0,1</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table> <p>n = 04 + DWL Where DWL indicates number of immediate data in terms of dwords.</p>	Default Value:	[20,68] Excludes DWORD 0,1	Format:	=n									
Default Value:	[20,68] Excludes DWORD 0,1													
Format:	=n													
1 BR13	<table border="1"> <tr> <td>31</td> <td>Reserved</td> </tr> <tr> <td colspan="2">Format: MBZ</td> </tr> </table>	31	Reserved	Format: MBZ										
	31	Reserved												
	Format: MBZ													
	<table border="1"> <tr> <td>30</td> <td>Clipping Enabled</td> </tr> <tr> <td colspan="2"> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table> </td> </tr> </table>	30	Clipping Enabled	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>		Value	Name	0b	Disabled	1b	Enabled			
	30	Clipping Enabled												
	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>		Value	Name	0b	Disabled	1b	Enabled						
Value	Name													
0b	Disabled													
1b	Enabled													
<table border="1"> <tr> <td>29:26</td> <td>Reserved</td> </tr> <tr> <td colspan="2">Format: MBZ</td> </tr> </table>	29:26	Reserved	Format: MBZ											
29:26	Reserved													
Format: MBZ														
<table border="1"> <tr> <td>25:24</td> <td>Color Depth</td> </tr> <tr> <td colspan="2"> <table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table> </td> </tr> </table>	25:24	Color Depth	<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>		Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
25:24	Color Depth													
<table border="1"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>		Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color			
Value	Name													
00b	8 Bit Color													
01b	16 Bit Color(565)													
10b	16 Bit Color(1555)													
11b	32 Bit Color													
<table border="1"> <tr> <td>23:16</td> <td>Raster Operation</td> </tr> <tr> <td>15:0</td> <td>Destination Pitch in DWords 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).</td> </tr> </table>	23:16	Raster Operation	15:0	Destination Pitch in DWords 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
23:16	Raster Operation													
15:0	Destination Pitch in DWords 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).													
2 BR22	<table border="1"> <tr> <td>31:16</td> <td>Destination Y1 Coordinate (Top) 16 bit signed number.</td> </tr> <tr> <td>15:0</td> <td>Destination X1 Coordinate (Left) 16 bit signed number.</td> </tr> </table>	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.	15:0	Destination X1 Coordinate (Left) 16 bit signed number.									
	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.												
15:0	Destination X1 Coordinate (Left) 16 bit signed number.													
3 BR23	<table border="1"> <tr> <td>31:16</td> <td>Destination Y2 Coordinate (Bottom) 16 bit signed number.</td> </tr> <tr> <td>15:0</td> <td>Destination X2 Coordinate (Right) 16 bit signed number.</td> </tr> </table>	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.	15:0	Destination X2 Coordinate (Right) 16 bit signed number.									
	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.												
15:0	Destination X2 Coordinate (Right) 16 bit signed number.													
4.5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	<table border="1"> <tr> <td>63:0</td> <td>Destination Base Address</td> </tr> <tr> <td colspan="2">Format: VIRTUAL_ADDR[63:0]</td> </tr> </table>	63:0	Destination Base Address	Format: VIRTUAL_ADDR[63:0]										
	63:0	Destination Base Address												
Format: VIRTUAL_ADDR[63:0]														

XY_PAT_BLT_IMMEDIATE		
6..n	31:0	Immediate Data

XY_PAT_BLT

XY_PAT_BLT										
Source:	BlitterCS									
Length Bias:	2									
<p>PAT_BLT is used when there is no source and the color pattern is not trivial (is not a solid color only). If clipping is enabled, all scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation. The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p>										
DWord	Bit	Description								
0 BR00	31:29	Client								
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode				
	Default Value:	02h 2D Processor								
	Format:	Opcode								
	28:22	Instruction Target(Opcode)								
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>51h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	51h	Format:	Opcode				
	Default Value:	51h								
	Format:	Opcode								
	21:20	32bpp Byte Mask This field is only used for 32bpp.								
		<table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td style="text-align: center;">[Default]</td> </tr> <tr> <td style="text-align: center;">1xb</td> <td>Write Alpha Channel</td> </tr> <tr> <td style="text-align: center;">x1b</td> <td>Write RGB Channel</td> </tr> </tbody> </table>	Value	Name	00b	[Default]	1xb	Write Alpha Channel	x1b	Write RGB Channel
Value		Name								
00b		[Default]								
1xb	Write Alpha Channel									
x1b	Write RGB Channel									
19:15	Reserved									
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
14:12	Pattern Horizontal Seed Pixel of the scan line to start on corresponding to DST X=0.									
11	Tiling Enable									
	<table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
	Value	Name	Description							
0b	Tiling Disabled (Linear Blit)									
1b	Tiling Enabled	Tile-X or Tile-Y.								
10:8	Pattern Vertical Seed Scan line of the 8x8 pattern to start on corresponding to DST Y=0.									
7:0	DWord Length									
	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Default Value:</td> <td>06h</td> </tr> </table>	Default Value:	06h							
Default Value:	06h									
1 BR13	31	Reserved								
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									

XY_PAT_BLT												
	30	Clipping Enabled <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled				
	Value	Name										
	0b	Disabled										
	1b	Enabled										
	29:26	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
	Format:	MBZ										
	25:24	Color Depth <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 70%;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
	Value	Name										
	00b	8 Bit Color										
	01b	16 Bit Color(565)										
10b	16 Bit Color(1555)											
11b	32 Bit Color											
23:16	Raster Operation											
15:0	Destination Pitch in DWords 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).											
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.										
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.										
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.										
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.										
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	Destination Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]											
6..7 28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The programmed Pattern Base Address must always be Cache Line (64byte) aligned.	63:0	Pattern Base Address <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]											

XY_PAT_CHROMA_BLT_IMMEDIATE

XY_PAT_CHROMA_BLT_IMMEDIATE										
Source:	BlitterCS									
Length Bias:	2									
<p>PAT_BLT_IMMEDIATE is used when there is no source and the color pattern is not trivial (is not a solid color only) and the pattern is pulled through the command stream. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns.</p> <p>DWL indicates the total number of Dwords of immediate data. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.</p> <p>The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p>										
DWord	Bit	Description								
0 BR00	31:29	Client <table border="1"> <tr> <td>Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode				
	Default Value:	02h 2D Processor								
	Format:	Opcode								
	28:22	Instruction Target(Opcode) <table border="1"> <tr> <td>Default Value:</td> <td>77h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	77h	Format:	Opcode				
	Default Value:	77h								
	Format:	Opcode								
	21:20	32bpp Byte Mask This field is only used for 32bpp. <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>[Default]</td> </tr> <tr> <td>1xb</td> <td>Write Alpha Channel</td> </tr> <tr> <td>x1b</td> <td>Write RGB Channel</td> </tr> </tbody> </table>	Value	Name	00b	[Default]	1xb	Write Alpha Channel	x1b	Write RGB Channel
	Value	Name								
	00b	[Default]								
	1xb	Write Alpha Channel								
x1b	Write RGB Channel									
19:17	Transparency Range Mode (chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)									
16:15	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
14:12	Pattern Horizontal Seed Pixel of the scan line to start on corresponding to DST X=0.									
11	Tiling Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td>1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
Value	Name	Description								
0b	Tiling Disabled (Linear Blit)									
1b	Tiling Enabled	Tile-X or Tile-Y.								

XY_PAT_CHROMA_BLT_IMMEDIATE

	10:8	Pattern Vertical Seed Scan line of the 8x8 pattern to start on corresponding to DST Y=0.									
	7:0	DWord Length <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Default Value:</td> <td>[22,70] Excludes DWORD 0,1</td> </tr> <tr> <td>Format:</td> <td>=n</td> </tr> </table> n = 06 + DWL Where DWL is immediate data pattern size in dwords.	Default Value:	[22,70] Excludes DWORD 0,1	Format:	=n					
Default Value:	[22,70] Excludes DWORD 0,1										
Format:	=n										
1 BR13	31	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
	30	Clipping Enabled <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Value</th> <th style="width: 50%;">Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled			
	Value	Name									
	0b	Disabled									
	1b	Enabled									
29:26	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 70%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
25:24	Color Depth <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Value</th> <th style="width: 70%;">Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name										
00b	8 Bit Color										
01b	16 Bit Color(565)										
10b	16 Bit Color(1555)										
11b	32 Bit Color										
23:16	Raster Operation										
15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.									
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.									
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.									
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.									
4.5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling	63:0	Destination Base Address <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]							
	Format:	VIRTUAL_ADDR[63:0]									

XY_PAT_CHROMA_BLT_IMMEDIATE

is not enabled, this address is to be CL (64byte) aligned.		
6 BR18	31:0	Transparency Color Low (Chroma-key Low = Pixel Greater or Equal)
7 BR19	31:0	Transparency Color High (Chroma-key High = Pixel Less or Equal)
8..n	31:0	Immediate Data

XY_PAT_CHROMA_BLT

XY_PAT_CHROMA_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>PAT_BLT is used when there is no source and the color pattern is not trivial (is not a solid color only). All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation. The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	76h
		Format:	Opcode
	21:20	32bpp Byte Mask This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
		1xb	Write Alpha Channel
		x1b	Write RGB Channel
19:17	Transparency Range Mode (chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)		
16:15	Reserved		
	Format:	MBZ	
14:12	Pattern Horizontal Seed Pixel of the scan line to start on corresponding to DST X=0.		
11	Tiling Enable		
	Value	Name	
	0b	Tiling Disabled (Linear Blit)	
1b	Tiling Enabled	Tile-X or Tile-Y.	
10:8	Pattern Vertical Seed Scan line of the 8x8 pattern to start on corresponding to DST Y=0.		
7:0	DWord Length		
	Default Value:	08h Excludes DWORD 0,1	
1	31	Reserved	

XY_PAT_CHROMA_BLT											
BR13		Format: MBZ									
	30	Clipping Enabled <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="width: 50%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Disabled</td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Disabled	1b	Enabled			
	Value	Name									
	0b	Disabled									
	1b	Enabled									
	29:26	Reserved Format: MBZ									
25:24	Color Depth <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%; text-align: center;">Value</th> <th style="width: 70%; text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00b</td> <td>8 Bit Color</td> </tr> <tr> <td style="text-align: center;">01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td style="text-align: center;">10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td style="text-align: center;">11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name										
00b	8 Bit Color										
01b	16 Bit Color(565)										
10b	16 Bit Color(1555)										
11b	32 Bit Color										
23:16	Raster Operation 15:0 Destination Pitch in DWords 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.									
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.									
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.									
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.									
4..5	63:0 Destination Base Address <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]										
6..7	63:0 Pattern Base Address <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table> (28:06 are implemented) (Note no NPO2 change here). The pattern data must be located in linear memory. The Pattern Base Address must always be Cache Line (64byte) aligned.	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]										
8 BR18	31:0	Transparency Color Low (Chroma-key Low = Pixel Greater or Equal)									
9 BR19	31:0	Transparency Color High (Chroma-key High = Pixel Less or Equal)									

XY_PIXEL_BLT

XY_PIXEL_BLT										
Source:	BlitterCS									
Length Bias:	2									
<p>The Destination X coordinate and Destination Y coordinate is compared with the ClipRect registers. If it is within all 4 comparisons, then the pixel supplied in the XY_SETUP_BLT instruction is written with the raster operation to (Destination Y Address + (Destination Y coordinate * Destination pitch) + (Destination X coordinate * bytes per pixel)).</p> <p>ROP field must specify pattern or fill with 0's or 1's. There is no source operand.</p> <p>Negative Stride (= Pitch) specified in the Setup command is Not Allowed</p>										
DWord	Bit	Description								
0 BR00	31:29	Client <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode				
	Default Value:	02h 2D Processor								
	Format:	Opcode								
	28:22	Instruction Target(Opcode) <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Default Value:</td> <td>24h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	24h	Format:	Opcode				
	Default Value:	24h								
	Format:	Opcode								
	21:12	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
Format:	MBZ									
11	Tiling Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
Value	Name	Description								
0b	Tiling Disabled (Linear Blit)									
1b	Tiling Enabled	Tile-X or Tile-Y.								
10:8	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
7:0	DWord Length <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Default Value:</td> <td>00h</td> </tr> </table>	Default Value:	00h							
Default Value:	00h									
1 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.								
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.								

XY_SCANLINES_BLT

XY_SCANLINES_BLT		
Source:	BlitterCS	
Length Bias:	2	
<p>All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation. The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8. Solid pattern should use the XY_SETUP_MONO_PATTERN_SL_BLT instruction. ROP field must specify pattern or fill with 0's or 1's. There is no source operand.</p>		
DWord	Bit	Description
0 BR00	31:29	Client
		Default Value: 02h 2D Processor
		Format: Opcode
	28:22	Instruction Target(Opcode)
		Default Value: 25h
		Format: Opcode
	21:15	Reserved
		Format: MBZ
14:12	Pattern Horizontal Seed	
	Pixel of the scan line to start on corresponding to DST X=0.	
11	Tiling Enable	
	Value	Name
	0b	Tiling Disabled (Linear Blit)
	1b	Tiling Enabled
10:8	Pattern Vertical Seed	
	Scan line of the 8x8 pattern to start on corresponding to DST Y=0.	
7:0	DWord Length	
	Default Value: 01h	
1 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.
2 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.

XY_SETUP_BLT

XY_SETUP_BLT		
Source:	BlitterCS	
Length Bias:	2	
<p>This setup instruction supplies common setup information including clipping coordinates used by the XY commands: XY_PIXEL_BLT, XY_SCANLINE_BLT, XY_TEXT_BLT, and XY_TEXT_BLT_IMMEDIATE. These are the only instructions that require that state be saved between instructions other than the Clipping parameters. There are 5 dedicated registers to contain the state for the 3 setup BLT instructions (XY_SETUP_BLT, XY_SETUP_MONO_PATTERN_SL_BLT, and XY_SETUP_CLIP_BLT). All other BLTs use a temporary version of these. The 5 double word registers are: DW1 (Setup Control), DW6 (Setup Foreground color), DW5 (Setup Background color), DW7 (Setup Pattern address), and DW4 (Setup Destination Base Address).</p>		
DWord	Bit	Description
0 BR00	31:29	Client
		Default Value: 02h 2D Processor
	Format: Opcode	
	28:22	Instruction Target(Opcode)
		Default Value: 01h
	Format: Opcode	
	21:20	32 bpp Byte Mask
		Value
1xb Write Alpha Channel		x1b Write RGB Channel
19:12	Reserved	
	Format: MBZ	
11	Tiling Enable	
	Value	Name
	0b Tiling Disabled (Linear Blit)	1b Tiling Enabled (Tile-X or Tile-Y)
10:8	Reserved	
	Format: MBZ	
7:0	DWord Length	
	Default Value: 08h	
1 BR01	31	Reserved
		Format: MBZ
	30	Clipping Enabled
Value	Name	
0b	Disabled	

XY_SETUP_BLT												
		<table border="1"> <tr> <td>1b</td> <td>Enabled</td> </tr> </table>	1b	Enabled								
1b	Enabled											
	29	Mono Source Transparency Mode <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0b</td> <td>Use Background</td> </tr> <tr> <td>1b</td> <td>Transparency Enabled</td> </tr> </tbody> </table>	Value	Name	0b	Use Background	1b	Transparency Enabled				
Value	Name											
0b	Use Background											
1b	Transparency Enabled											
	28:26	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ											
	25:24	Color Depth <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </tbody> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name											
00b	8 Bit Color											
01b	16 Bit Color(565)											
10b	16 Bit Color(1555)											
11b	32 Bit Color											
	23:16	Raster Operation										
	15:0	Destination Pitch in DWords 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR24	31:16	ClipRect Y1 Coordinate (Top) (30:16 = 15 bit positive number)										
	15:0	ClipRect X1 Coordinate (Left) (14:00 = 15 bit positive number)										
3 BR25	31:16	ClipRect Y2 Coordinate (Bottom) (30:16 = 15 bit positive number)										
	15:0	ClipRect X2 Coordinate (Right) (14:00 = 15 bit positive number)										
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	Destination Base Address <table border="1"> <tr> <td>Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]											
6 BR05	31:0	Setup Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All										
7 BR06	31:0	Setup Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)										
8..9 28:06 are implemented. Note no NPO2 change here. The pattern data must be	63:0	Setup Pattern Base Address for Color Pattern <table border="1"> <tr> <td>Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]								
Format:	VIRTUAL_ADDR[63:0]											

XY_SETUP_BLT

located in linear memory. The Setup Pattern Base Address for Color Pattern must always be Cache Line (64byte) aligned.		
--	--	--

XY_SETUP_CLIP_BLT

XY_SETUP_CLIP_BLT		
Source:	BlitterCS	
Length Bias:	2	
This command is used to only change the clip coordinate registers. These are the same clipping registers as the Setup clipping registers above.		
DWord	Bit	Description
0 BR00	31:29	Client
		Default Value: 02h 2D Processor
		Format: Opcode
	28:22	Instruction Target(Opcod
		Default Value: 03h
		Format: Opcode
	21:12	Reserved
Format: MBZ		
11	Tiling Enable	
	Value	Name
	0b	Tiling Disabled (Linear Blit)
	1b	Tiling Enabled (Tile-X or Tile-Y
	10:8	Reserved
Format: MBZ		
7:0	DWord Length	
	Default Value: 01h	
1 BR24	31:16	ClipRect Y1 Coordinate (Top) (30:16 = 15 bit positive number)
	15:0	ClipRect X1 Coordinate (Left) (14:00 = 15 bit positive number)
2 BR25	31:16	ClipRect Y2 Coordinate (Bottom) (30:16 = 15 bit positive number)
	15:0	ClipRect X2 Coordinate (Right) (14:00 = 15 bit positive number)

XY_SETUP_MONO_PATTERN_SL_BLT

XY_SETUP_MONO_PATTERN_SL_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>This setup instruction supplies common setup information including clipping coordinates used exclusively with the following instruction: XY_SCANLINE_BLT (SLB) - 1 scan line of monochrome pattern and destination are the only operands allowed.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	11h
		Format:	Opcode
	21:20	32 bpp Byte Mask	
		Value	Name
1xb		Write Alpha Channel	
	x1b	Write RGB Channel	
19:12	Reserved		
	Format:	MBZ	
11	Tiling Enable		
	Value	Name	
	0b	Tiling Disabled (Linear Blit)	
	1b	Tiling Enabled (Tile-X or Tile-Y)	
10:8	Reserved		
	Format:	MBZ	
7:0	DWord Length		
	Default Value:	08h	
1 BR01	31	Solid Pattern Select (SLB and Pixel only)	
		Value	Name
		0	No Solid Pattern
		1	Solid Pattern
	30	Clipping Enabled	
Value		Name	
	0b	Disabled	

XY_SETUP_MONO_PATTERN_SL_BLT

		1b	Enabled
	29	Reserved	
		Format:	MBZ
	28	Mono Pattern Transparency Mode	
		Value	Name
		0b	Use Background
		1b	Transparency Enabled
	27:26	Reserved	
		Format:	MBZ
	25:24	Color Depth	
		Value	Name
		00b	8 Bit Color
		01b	16 Bit Color(565)
		10b	16 Bit Color(1555)
		11b	32 Bit Color
	23:16	Raster Operation	
	15:0	Destination Pitch in DWords 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).	
2 BR24	31:16	ClipRect Y1 Coordinate (Top) (30:16 = 15 bit positive number)	
	15:0	ClipRect X1 Coordinate (Left) (14:00 = 15 bit positive number)	
3 BR25	31:16	ClipRect Y2 Coordinate (Bottom) (30:16 = 15 bit positive number)	
	15:0	ClipRect X2 Coordinate (Right) (14:00 = 15 bit positive number)	
4..5 This bitfield contains the base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	Setup Destination Base Address	
		Format:	VIRTUAL_ADDR[63:0]
6 BR05	31:0	Setup Background Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All	
7 BR06	31:0	Setup Foreground Color 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)	

XY_SETUP_MONO_PATTERN_SL_BLT		
8 BR20	31:0	DW0 (least significant) for a Monochrome Pattern
9 BR21	31:0	DW1 (most significant) for a Monochrome Pattern



XY_SRC_COPY_BLT

XY_SRC_COPY_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>This BLT instruction performs a color source copy where the only operands involved is a color source and destination of the same bit width.</p> <p>The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. The ROP value chosen must involve source and no pattern data in the ROP operation.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value: 02h 2D Processor	
		Format: Opcode	
	28:22	Instruction Target(Opcode)	
		Default Value: 53h	
		Format: Opcode	
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
	x1b	Write RGB Channel	
19:16	Reserved		
	Format: MBZ		
15	Src Tiling Enable		
	Value	Name	Description
	0b	Tiling Disabled (Linear)	
	1b	Tiling Enabled	Tile-X or Tile-Y.
14:12	Reserved		
	Format: MBZ		
11	Dest Tiling Enable		
	Value	Name	Description
	0b	Tiling Disabled (Linear Blit)	

XY_SRC_COPY_BLT											
		<table border="1"> <tr> <td>1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </table>	1b	Tiling Enabled	Tile-X or Tile-Y.						
	1b	Tiling Enabled	Tile-X or Tile-Y.								
	10:8	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ										
7:0	DWord Length <table border="1"> <tr> <td>Format:</td> <td>=n</td> </tr> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td>08h</td> <td></td> </tr> </table>	Format:	=n	Value	Name	08h					
Format:	=n										
Value	Name										
08h											
1 BR13	31	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
	Format:	MBZ									
	30	Clipping Enabled <table border="1"> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td>0b</td> <td>Disabled</td> </tr> <tr> <td>1b</td> <td>Enabled</td> </tr> </table>	Value	Name	0b	Disabled	1b	Enabled			
	Value	Name									
	0b	Disabled									
	1b	Enabled									
29:26	Reserved <table border="1"> <tr> <td>Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ								
Format:	MBZ										
25:24	Color Depth <table border="1"> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> <tr> <td>00b</td> <td>8 Bit Color</td> </tr> <tr> <td>01b</td> <td>16 Bit Color(565)</td> </tr> <tr> <td>10b</td> <td>16 Bit Color(1555)</td> </tr> <tr> <td>11b</td> <td>32 Bit Color</td> </tr> </table>	Value	Name	00b	8 Bit Color	01b	16 Bit Color(565)	10b	16 Bit Color(1555)	11b	32 Bit Color
Value	Name										
00b	8 Bit Color										
01b	16 Bit Color(565)										
10b	16 Bit Color(1555)										
11b	32 Bit Color										
23:16	Raster Operation <table border="1"> <tr> <td>15:0</td> <td> Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords). </td> </tr> </table>	15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).								
15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).										
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.									
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.									
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.									
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.									
4.5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled),	63:0	Destination Base Address <table border="1"> <tr> <td>Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]							
Format:	VIRTUAL_ADDR[63:0]										

XY_SRC_COPY_BLT				
this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.				
6 BR26	31:16	Source Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Source X1 Coordinate (Left) 16 bit signed number.		
7 BR11	31:16	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="width: 50px; text-align: center;">MBZ</td></tr></table>		MBZ
		MBZ		
15:0	Source Pitch (double word aligned) and in DWords 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).			
8..9 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	Source Base Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="width: 150px; text-align: center;">VIRTUAL_ADDR[63:0]</td></tr></table>		VIRTUAL_ADDR[63:0]
			VIRTUAL_ADDR[63:0]	

XY_SRC_COPY_CHROMA_BLT

XY_SRC_COPY_CHROMA_BLT			
Source:	BlitterCS		
Length Bias:	2		
<p>This BLT instruction performs a color source copy with chroma-keying where the only operands involved is a color source and destination of the same bit width.</p> <p>The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. The ROP value chosen must involve source and no pattern data in the ROP operation.</p>			
DWord	Bit	Description	
0 BR00	31:29	Client	
		Default Value:	02h 2D Processor
		Format:	Opcode
	28:22	Instruction Target(Opcode)	
		Default Value:	73h
		Format:	Opcode
	21:20	32bpp Byte Mask	
		This field is only used for 32bpp.	
		Value	Name
		00b	[Default]
1xb		Write Alpha Channel	
19:17	Transparency Range Mode		
	(chroma-key)		
16	Reserved		
	Format:	MBZ	
15	Src Tiling Enable		
	Value	Name	Description
	0b	Tiling Disabled (Linear)	
	1b	Tiling Enabled	Tile-X or Tile-Y.
14:12	Reserved		
	Format:	MBZ	
11	Dest Tiling Enable		

XY_SRC_COPY_CHROMA_BLT

		Value	Name	Description	
		0b	Tiling Disabled (Linear Blit)		
		1b	Tiling Enabled	Tile-X or Tile-Y.	
		Reserved			
		Format:		MBZ	
		DWord Length			
		Value	Name		
		0Ah			
1 BR13	31	Reserved			
			Format: MBZ		
	30	Clipping Enabled			
			Value	Name	
			0b	Disabled	
			1b	Enabled	
			Reserved		
		Format:		MBZ	
		Color Depth			
		Value	Name		
		00b	8 Bit Color		
		01b	16 Bit Color(565)		
		10b	16 Bit Color(1555)		
		11b	32 Bit Color		
		Raster Operation			
		15:0	Destination Pitch in DWords 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).		
2 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.			
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.			
3 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.			
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.			
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled),	63:0	Destination Base Address			
			Format:	VIRTUAL_ADDR[63:0]	

XY_SRC_COPY_CHROMA_BLT				
this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.				
6 BR26	31:16	Source Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Source X1 Coordinate (Left) 16 bit signed number.		
7 BR11	31:16	Reserved Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="width: 100px;">MBZ</td></tr></table>		MBZ
		MBZ		
15:0	Source Pitch (double word aligned) and in DWords 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).			
8..9 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	Source Base Address Format: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="width: 100px;"></td><td style="width: 100px;">VIRTUAL_ADDR[63:0]</td></tr></table>		VIRTUAL_ADDR[63:0]
	VIRTUAL_ADDR[63:0]			
10 BR18	31:0	Transparency Color Low (Chroma-key Low = Pixel Greater or Equal)		
11 BR19	31:0	Transparency Color High (Chroma-key High = Pixel Less or Equal)		

XY_TEXT_BLT

XY_TEXT_BLT										
Source:	BlitterCS									
Length Bias:	2									
<p>All source scan lines and pixels that fall within the ClipRect Y and X coordinates are written. The source address corresponds to Destination X1 and Y1 coordinate.</p> <p>Text is either bit or byte packed. Bit packed means that the next scan line starts 1 pixel after the end of the current scan line with no bit padding. Byte packed means that the next scan line starts on the first bit of the next byte boundary after the last bit of the current line.</p> <p>Source expansion color registers are always in the SETUP_BLT.</p> <p>Negative Stride (= Pitch) is NOT ALLOWED.</p>										
DWord	Bit	Description								
0 BR00	31:29	Client <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode				
	Default Value:	02h 2D Processor								
	Format:	Opcode								
	28:22	Instruction Target(Opcode) <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Default Value:</td> <td>26h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	26h	Format:	Opcode				
	Default Value:	26h								
	Format:	Opcode								
	21:17	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
	16	Bit / Byte Packed Byte packed is for the NT driver. <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 30%;">Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Bit</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Byte</td> </tr> </tbody> </table>	Value	Name	0	Bit	1	Byte		
	Value	Name								
0	Bit									
1	Byte									
15:12	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
11	Tiling Enable <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 15%;">Value</th> <th style="width: 40%;">Name</th> <th style="width: 45%;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
Value	Name	Description								
0b	Tiling Disabled (Linear Blit)									
1b	Tiling Enabled	Tile-X or Tile-Y.								
10:8	Reserved <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
7:0	DWord Length <table border="1" style="width: 100%;"> <tr> <td style="width: 30%;">Default Value:</td> <td>03h</td> </tr> </table>	Default Value:	03h							
Default Value:	03h									

XY_TEXT_BLT				
1 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.		
2 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.		
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.		
3.4 Address of the first byte on a scan line corresponding to source X1, Y1. Note no NPO2 change here. The source address must always be Cache Line (64byte) aligned.	63:0	Source Address <table border="1" style="width: 100%;"> <tr> <td style="width: 15%;">Format:</td> <td>VIRTUAL_ADDR[63:0]</td> </tr> </table>	Format:	VIRTUAL_ADDR[63:0]
Format:	VIRTUAL_ADDR[63:0]			

XY_TEXT_IMMEDIATE_BLT

XY_TEXT_IMMEDIATE_BLT										
Source:	BlitterCS									
Length Bias:	2									
<p>This instruction allows the Driver to send data through the instruction stream that eliminates the read latency of reading a source from memory.</p> <p>If an operand is in system cacheable memory and either small or only accessed once, it can be copied directly to the instruction stream versus to graphics accessible memory. The IMMEDIATE_BLT data MUST transfer an even number of doublewords.</p> <p>The BLT engine will hang if it does not get an even number of doublewords. All source scan lines and pixels that fall within the ClipRect X and Y coordinates are written. The source data corresponds to Destination X1 and Y1 coordinate.</p> <p>Source expansion color registers are always in the SETUP_BLT. NEGATIVE STRIDE (= PITCH) IS NOT ALLOWED.</p>										
DWord	Bit	Description								
0 BR00	31:29	Client <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>02h 2D Processor</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	02h 2D Processor	Format:	Opcode				
	Default Value:	02h 2D Processor								
	Format:	Opcode								
	28:22	Instruction Target(Opcode) <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>31h</td> </tr> <tr> <td>Format:</td> <td>Opcode</td> </tr> </table>	Default Value:	31h	Format:	Opcode				
	Default Value:	31h								
	Format:	Opcode								
	21:17	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ						
	Format:	MBZ								
	16	Bit / Byte Packed Byte packed is for the NT driver. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Value</th> <th style="text-align: center;">Name</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>Bit</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Byte</td> </tr> </tbody> </table>	Value	Name	0	Bit	1	Byte		
	Value	Name								
0	Bit									
1	Byte									
15:12	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
11	Tiling Enable <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%; text-align: center;">Value</th> <th style="width: 55%; text-align: center;">Name</th> <th style="width: 30%; text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0b</td> <td>Tiling Disabled (Linear Blit)</td> <td></td> </tr> <tr> <td style="text-align: center;">1b</td> <td>Tiling Enabled</td> <td>Tile-X or Tile-Y.</td> </tr> </tbody> </table>	Value	Name	Description	0b	Tiling Disabled (Linear Blit)		1b	Tiling Enabled	Tile-X or Tile-Y.
Value	Name	Description								
0b	Tiling Disabled (Linear Blit)									
1b	Tiling Enabled	Tile-X or Tile-Y.								
10:8	Reserved <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Format:</td> <td>MBZ</td> </tr> </table>	Format:	MBZ							
Format:	MBZ									
7:0	DWord Length <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">Default Value:</td> <td>[17,65] Excludes DWORD 0,1</td> </tr> </table>	Default Value:	[17,65] Excludes DWORD 0,1							
Default Value:	[17,65] Excludes DWORD 0,1									

XY_TEXT_IMMEDIATE_BLT				
		<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Format:</td> <td>=n</td> </tr> </table> <p>n = 01 + DWL Where DWL indicates size of indirect data in dwords.</p>	Format:	=n
Format:	=n			
1 BR22	31:16	Destination Y1 Coordinate (Top) 16 bit signed number.		
	15:0	Destination X1 Coordinate (Left) 16 bit signed number.		
2 BR23	31:16	Destination Y2 Coordinate (Bottom) 16 bit signed number.		
	15:0	Destination X2 Coordinate (Right) 16 bit signed number.		
3..n	31:0	Immediate Data		