VESA STANDARDS CHANGE REQUEST FORM

To be Filled in by Submitter (Refer to VESA Document VP235H, Section 5)

TITLE:	DSC 1.1 fractional bpp underflow SCR
AFFECTED DOCUMENT:	DSC 1.1 specification (including dsc_model_20140715.zip)
REVISION CATEGORY:	Category 1 (Refer to VP235H Appendix A; will be subject to Task Group review)
SUBMITTED TO:	Task Group
SPONSOR:	Fred Walls, Broadcom

SCR REVISION HISTORY	
(DATE)	**(CHANGE)**
05/19/2015	**Initial Submission of SCR**
05/28/2015 v2	**Modifications in response to TG feedback** - **Request for modified PPM read function for better compatibility**

(add more rows as needed)

To be Filled in by VESA Office:

VESA SCR NUMBER:	DSC 1.1 fractional bpp underflow SCR
SCR ENTRY DATE:	5/19/15

To be Filled in by Task Group or VESA Office

SCR **ADOPTED, REJECTED, or otherwise DISPOSITIONED for other action**	SCR is (adopted) or (rejected) or (Dispositioned for other action) If rejected, explain reason for acceptation or rejection If dispositioned, explain action or plan for action (such as including in future draft specification revision, or re-visiting at future date, or other)
DATE SCR ADOPTED	Sept. 3, 2015

Summary of the Proposed Change(s)

This change fixes an underflow condition that occurs with certain combinations of slice width and fractional bits/pixel. This change allows these modes to supported correctly.

Change section 6.8.1 to include updated pseudocode for adjustment bits

DSC model – change VLCGroup() function to include updated code for adjustment bits

IPR (Intellectual Property Rights) declaration, if any

The submitter must be familiar with VESA Policy 200B. If an IPR declaration is to be made:

> Submitter must provide the declaration in writing to VESA as per section 4.2 of VESA Policy 200B.
>
> The published Standard Revision or Errata document will include the IPR holder name, contact information, and claims known, in keeping with VESA specification standards.

Benefits as a Result of the Changes

Certain combinations of slice width and fractional bpp currently cause the model to fail with an underflow error message. This change fixes these failures.

Assessment of the Impact

Modes that currently fail will now be supported. Currently working modes are unaffected.

Analysis of the Device Hardware Implication

No impact. Current hardware cannot support these modes, so future hardware may support them with the change.

Analysis of the Device Software Implications

No impact.

Analysis of the Compliance Test & Interop Implications

TG could modify CTG to add a test of one of the bad modes to ensure the change is implemented correctly. No interoperability issues since the change does not affect modes that currently work.

New Referenced Documents Resulting from Change

N/A

Attachments

dsc_codec.c – Includes code update that is needed

Proposed Document Change(s) or Addition(s)

6.8.1 Buffer Level Tracker

…

The **forceMpp** value applies to the group immediately prior to the one that coincides with the end of a chunk:

```
bugFixCondition = (bits_per_pixel * slice_width) & 0xf;

if((numBitsChunk + maxBitsPerGroup + 8 > 8 * chunk_size) ||
   (bugFixCondition && numBitsChunk + maxBitsPerGroup + 8 == 8 *
chunk_size)
            forceMpp = (pixelCount >
             initial_xmit_delay) && (bufferFullness - 8
             < maxBitsPerGroup -  pixelsPerGroup);
```

This ensures that there is always a sufficient number of bits in the encoder buffer to output the stuffed "0" padding bits.

…

dsc_codec.c:

```
void VLCGroup(dsc_cfg_t *dsc_cfg, dsc_state_t *dsc_state, unsigned char **byte_out_p)
{
	int i;
	int start_fullness[NUM_COMPONENTS];
	int adjFullness;
	int maxBitsPerGroup;
	int bugFixCondition;

	for (i=0; i<NUM_COMPONENTS; ++i)
	{
		dsc_state->midpointSelected[i] = 0;
		start_fullness[i] = dsc_state->encBalanceFifo[i].fullness;
	}

	// 444; Unit is same as CType
	dsc_state->prevNumBits = dsc_state->numBits;

	// Check stuffing condition
	dsc_state->forceMpp = 0;
	// Force MPP mode if buffer fullness is low
	//  Buffer threshold is ceil(bpp * 3) - 3, the first term is how many
	//   bits are removed from the model, the second term (3) is the minimum
	//   number of bits that a group can be coded with
	maxBitsPerGroup = (3 * dsc_cfg->bits_per_pixel + 15) >> 4;
	adjFullness = dsc_state->bufferFullness;
	bugFixCondition = (dsc_cfg->bits_per_pixel * dsc_cfg->slice_width) & 0xf;   //
Fractional bit left at end of slice
	if( (bugFixCondition && (dsc_state->numBitsChunk + maxBitsPerGroup + 8 == dsc_cfg-
>chunk_size * 8)) ||
        (dsc_state->numBitsChunk + maxBitsPerGroup + 8 > dsc_cfg->chunk_size * 8))
~~    if(dsc_state->numBitsChunk + maxBitsPerGroup + 8 > dsc_cfg->chunk_size * 8)~~
    {
        // End of chunk check to see if there is a potential to underflow
        // assuming adjustment bits are sent.
        adjFullness -= 8;
        if (adjFullness  < maxBitsPerGroup - 3)  // Force MPP is possible in VBR only
at end of line to pad chunks to byte boundaries
            dsc_state->forceMpp = 1;
    }
    else if((!dsc_cfg->vbr_enable) && (dsc_state->pixelCount >= dsc_cfg-
>initial_xmit_delay))  // underflow isn't possible if we're not removing bits
    {
        if (adjFullness  < maxBitsPerGroup - 3)
            dsc_state->forceMpp = 1;
    }

    for (i=0; i<NUM_COMPONENTS; ++i)
        VLCUnit(dsc_cfg, dsc_state, i, dsc_state->quantizedResidual[i]);

    // Keep track of fullness for each coded unit in the balance FIFO's
    for (i=0; i<NUM_COMPONENTS; ++i)
        fifo_put_bits(&(dsc_state->seSizeFifo[i]), dsc_state-
>encBalanceFifo[i].fullness - start_fullness[i], 6);

    if (dsc_cfg->muxing_mode == 0)  // Write data immedately to buffer
        WriteEntryToBitstream(dsc_cfg, dsc_state, *byte_out_p);
    else if (dsc_cfg->muxing_mode)  // substream muxing
    {
        //if (dsc_state->groupCount > dsc_cfg->mux_word_size + MAX_SE_SIZE - 1)
        if (dsc_state->groupCount > dsc_cfg->mux_word_size + MAX_SE_SIZE - 3)
            ProcessGroupEnc(dsc_cfg, dsc_state, *byte_out_p);
    }

    dsc_state->bufferFullness += dsc_state->numBits - dsc_state->prevNumBits;
    if ( dsc_state->bufferFullness > dsc_cfg->rcb_bits ) {
        // This check may actually belong after tgt_bpg has been subtracted
        printf("The buffer model has overflowed.  This probably occurred due to an
error in the\n");
        printf("rate control parameter programming.\n\n");
        printf( "ERROR: RCB overflow; size is %d, tried filling to %d\n", dsc_cfg-
>rcb_bits, dsc_state->bufferFullness );
        exit(1);
    }
    dsc_state->codedGroupSize = dsc_state->numBits - dsc_state->prevNumBits;

    dsc_state->prevMasterQp = dsc_state->masterQp;
    dsc_state->groupCountLine++;
}

…
```

utl.c: (replace read_ppm() function with the following)

```
/*! \param fp      Pointer to open file handle
    \param token   Storage for token
       \param line    Current line of data (modified if new line encountered)
       \param pos       Position in line (modified)
    \return        Picture loaded from file */
void gettoken(FILE *fp, char *token, char *line, int *pos)
{
       char c;
       int count = 0;

       // Get whitespace
       c = line[(*pos)++];
       while((c=='\0')||(c==' ')||(c=='\t')||(c==10)||(c==13)||(c=='#'))
       {
              if(c=='\0' || c=='\n' || c=='#')
              {
                     fgets(line, 1000, fp);
                     *pos = 0;
              }
              c = line[(*pos)++];
       }

       // Get token
       while(count<999 && !((c=='\0')||(c==' ')||(c=='\t')||(c==10)||(c==13)||(c=='#')))
       {
              token[count++] = c;
              c = line[(*pos)++];
       }
       token[count] = '\0';
}


//! Read PPM (portable pix map) file
/*! \param fp      Pointer to open file handle
    \return        Picture loaded from file */
pic_t *readppm(FILE *fp)
{
    pic_t *p;

    char magicnum[128];
    char line[1000];
       char token[1000];

    int w, h;
    int i, j;
    int g;
    int maxval;
       int pos = 0;

       fgets(line, 1000, fp);
       gettoken(fp, token, line, &pos);

    if (token[0] != 'P')
    {
        Err("Incorrect file type.");
    }
```

```
       strcpy(magicnum, token);

       gettoken(fp, token, line, &pos);
    w = atoi(token);
       gettoken(fp, token, line, &pos);
    h = atoi(token);
       gettoken(fp, token, line, &pos);
       maxval = atoi(token);

    p = pcreate(FRAME, RGB, YUV_444, w, h);
    if(maxval <= 255)
              p->bits = 8;
    else if(maxval <= 1023)
              p->bits = 10;
    else if(maxval <= 4095)
              p->bits = 12;
       else if(maxval <= 16383)
              p->bits = 14;
    else if(maxval <= 65535)
              p->bits = 16;
    else
    {
              printf("PPM read error, maxval = %d\n", maxval);
              pdestroy(p);
              return(NULL);
    }

    if (magicnum[1] == '2')
        for (i = 0; i < h; i++)
            for (j = 0; j < w; j++)
            {
                fscanf(fp, "%d", &g);  // Gray value in PGM
                p->data.rgb.r[i][j] = g;
                p->data.rgb.g[i][j] = g;
                p->data.rgb.b[i][j] = g;
            }
    else if (magicnum[1] == '3')
       {
              int c, v;
              i = 0; j = 0; c = 0;
              while( i < h )
              {
                     char *rest;
                     do
                     {
                            int sz = -1;
                            do
                            {
                                   sz++;
                                   line[sz] = fgetc(fp);
                                   if(feof(fp))
                                          line[++sz] = '\n';
                            } while((line[sz] != '\n') && ((sz < 900) || ((line[sz] >= '0')
&& (line[sz] <= '9'))));
                            line[sz+1] = '\0';
                     } while (line[0] == '#');
                     rest = line;
                     while((rest[0] != '\0') && ((rest[0] < '0') || (rest[0] > '9'))) rest++;
                     while(rest[0] != '\0')
                     {
                            v = 0;
                            while((rest[0] >= '0') && (rest[0] <= '9'))
                            {
                                   v = 10*v + (rest[0] - '0');
                                   rest++;
                            }
                            if(c==0)       p->data.rgb.r[i][j] = v;
                            else if(c==1) p->data.rgb.g[i][j] = v;
                            else if(c==2) p->data.rgb.b[i][j] = v;
                            c++;
                            if(c>2)
                            {
                                   c = 0; j++;
                                   if(j>=w)
                                   {
                                          j = 0; i++;
                                   }
                            }
                            while((rest[0] != '\0') && ((rest[0] < '0') || (rest[0] > '9')))
rest++;
                     }
              }
    }
    else if (magicnum[1] == '5') // PGM binary
        for (i = 0; i < h; i++)
            for (j = 0; j < w; j++)
            {
              g = (unsigned char)fgetc(fp);  // Gray value
              if(maxval > 255)
                  g = (g<<8) + (unsigned char)fgetc(fp);
                p->data.rgb.r[i][j] = g;
                p->data.rgb.g[i][j] = g;
                p->data.rgb.b[i][j] = g;
            }
    else // P6
       {
        for (i = 0; i < h; i++)
              {
            for (j = 0; j < w; j++)
            {
                p->data.rgb.r[i][j] = (unsigned char) fgetc(fp);
                            if(maxval > 255)
                                   p->data.rgb.r[i][j] = (p->data.rgb.r[i][j] << 8) +
(unsigned char)fgetc(fp);
                            p->data.rgb.g[i][j] = (unsigned char) fgetc(fp);
                            if(maxval > 255)
                                   p->data.rgb.g[i][j] = (p->data.rgb.g[i][j] << 8) +
(unsigned char)fgetc(fp);
                            p->data.rgb.b[i][j] = (unsigned char) fgetc(fp);
                            if(maxval > 255)
                                   p->data.rgb.b[i][j] = (p->data.rgb.b[i][j] << 8) +
(unsigned char)fgetc(fp);
            }
              }
       }

    return p;
}
```

Background Information

Pixelworks reported that certain combinations of fractional bpp and slice width can cause the software to prematurely terminate with a reported underflow condition. Since the C model is normative, these combinations are effectively unsupported by DSC 1.1. This SCR seeks to allow those modes to be used without affecting modes that currently work.

- End of Document -

VESA STANDARDS CHANGE REQUEST FORM

To be Filled in by Submitter (Refer to VESA Document VP235H, Section 5)

TITLE:	DSC 1.1 PPS guidance SCR
AFFECTED DOCUMENT:	DSC 1.1 specification (including dsc_model_20140715.zip)
REVISION CATEGORY:	Category 1 (Refer to VP235H Appendix A; will be subject to Task Group review)
SUBMITTED TO:	Task Group
SPONSOR:	Fred Walls, Broadcom

SCR REVISION HISTORY	
(DATE)	**(CHANGE)**
05/19/2015	**Initial Submission of SCR**
05/28/2015 v2	**Incorporate TGR feedback:** - **Fix formula for first_line_bpg_offset and incorporate in codec_main.c** - **Incorporate updated PPM read routine in software**
07/13/2015 v3	**Incorporate GMR feedback:** - **Update slice size guidance for Table E-3**
09/14/2015 v4	**Incorporate post-GMR feedback from MIPI:** - **Reduce INITIAL_DELAY and FIRST_LINE_BPG_OFFSET to reduce buffer requirement** **Merge C model with underflow fix SCR (adopted) C model**

(add more rows as needed)

To be Filled in by VESA Office:

VESA SCR NUMBER:	DSC 1.1 PPS guidance SCR
SCR ENTRY DATE:	5/19/15

To be Filled in by Task Group or VESA Office

SCR **ADOPTED, REJECTED, or otherwise DISPOSITIONED for other action**	**SCR is (adopted)** or (rejected) or (Dispositioned for other action) If rejected, explain reason for acceptation or rejection If dispositioned, explain action or plan for action (such as including in future draft specification revision, or re-visiting at future date, or other)
DATE SCR ADOPTED	10/1/15

Summary of the Proposed Change(s)

This change involves changing the guidance that is provided in the standard for PPS parameters.

Annex E – Derivation of Rate Control Parameters (Informative)

Table E-2 – guidance for ***first_line_bpg_offset*** and ***initial_xmit_delay***

Table E-5 – parameters to match those that were tested for PPS update

DSC C model – Add recommended parameters with tall slices. Keep previous parameter sets in a legacy directory.

IPR (Intellectual Property Rights) declaration, if any

The submitter must be familiar with VESA Policy 200B. If an IPR declaration is to be made:

Submitter must provide the declaration in writing to VESA as per section 4.2 of VESA Policy 200B.

The published Standard Revision or Errata document will include the IPR holder name, contact information, and claims known, in keeping with VESA specification standards.

Benefits as a Result of the Changes

The new parameters help reduce scintillation on certain types of moving test images.

Assessment of the Impact

The new parameters simply represent new guidance for implementers. There is no normative change.

Analysis of the Device Hardware Implication

For most implementations, no hardware change is needed. Some implementations with hard-coded PPS or rate buffer size limitations may require slight modifications in order to take advantage of the updated guidance.

Analysis of the Device Software Implications

The software for devices may be updated to take advantage of the new guidance. The old parameters still comply to the specification.

Analysis of the Compliance Test & Interop Implications

The CTG could be updated to test devices using the updated guidance; however, this is not entirely necessary since the CTG already provides full test coverage.

Transport specifications that allow communication of the rate buffer model size should not have any interoperability issues, since this allows the PPS values to be negotiated. Transport specifications that adopt the updated guidance should exercise care to ensure interoperability is maintained.

New Referenced Documents Resulting from Change

N/A

Attachments

dsc_model_20150914.zip – C model that includes updated PPS parameter config files, PPM read fix, and code to automatically select ***first_line_bpg_offset*** if unspecified. Updated to include previous underflow fix SCR.

Proposed Document Change(s) or Addition(s)

Table E-2 lists recommended and required PPS syntax element rate control values.

Table E-2: Recommended and Required PPS Syntax Element Rate Control Values

PPS Syntax Element	Recommended and Required Values
first_line_bpg_offset	The first line of each slice does not code as efficiently as subsequent lines, due to the lack of prediction and Indexed Color History (ICH) upper neighboring pixels. To maintain uniform visual quality across a slice, it is important to provide an extra bit allocation for the first line. Empirical results have shown that a value of ~~12bpg~~15bpg works well in general. ~~at 8bpp, and 15bpg works well at 12bpp.~~The ***first_line_bpg_offset*** value should be smaller when ***slice_height*** is smaller, so it is recommended that it be scaled according to ***slice_height***: ***first_line_bpg_offset*** = 12 + (int)(0.09 * MIN(34, ***slice_height*** – 8)) for ***slice_height*** >= 8, and ***first_line_bpg_offset*** = 2 * (***slice_height*** – 1) for ***slice_height*** < 8.
initial_xmit_delay	If the initial transmission delay is 0, the buffer level would need to be constrained to a "0" bit at the end of a slice to guarantee that a slice contains the correct number of bits. This could be problematic because it would be difficult to ensure good visual quality at the end of a slice. A non-zero ***initial_xmit_delay*** allows a final maximum buffer fullness of up to ***initial_xmit_delay*** * ***bits_per_pixel***. Empirical results have shown ~~that an optimal value satisfies~~good performance when ***initial_xmit_delay*** * ***bits_per_pixel*** ≈ ***rc_model_size*** * (1 / 2) ~~/ 2~~.
…	…

Table E-3: Recommended Alternative Slice Dimensions to Prevent *scale_increment_interval*

Problem Configuration	Problem Slice Dimensions	Recommended Slice Dimensions
Default RC parameters, 8bpp	2048x4096	2048x~~1024~~2048
Default RC parameters, 8bpp	1024x4096	1024x2048
Default RC parameters, 8bpp	4096x2048	4096x1024
Default RC parameters, 12bpp	2048x4096	2048x2048

…

Table E-5: Selected Rate Control-Related Parameter Recommended Values[a]

Syntax Element[b]	At 8bpp/ 8bpc	At 8bpp/ 10bpc	At 8bpp/ 12bpc	At 12bpp/ 8bpc	At 12bpp/ 10bpc	At 12bpp/ 12bpc
initial_xmit_delay	512	512	512	341	341	341
first_line_bpg_offset	~~12~~15	~~12~~15	~~12~~15	15	15	15
initial_offset	6144	6144	6144	2048	2048	2048
flatness_min_qp	3	7	11	3	7	11
flatness_max_qp	12	16	20	12	16	20

rc_quant_incr_limit0	11	15	19	11	15	19
rc_quant_incr_limit1	11	15	19	11	15	19
rc_range_parameters[0]	MinQp: 0 MaxQp: 4 Ofs: 2	MinQp: 0 MaxQp: ~~4~~8 Ofs: 2	MinQp: 0 MaxQp: 12 Ofs: 2	MinQp: 0 MaxQp: 2 Ofs: 2	MinQp: 0 MaxQp: 2 Ofs: 2	MinQp: 0 MaxQp: 6 Ofs: 2
rc_range_parameters[1]	MinQp: 0 MaxQp: 4 Ofs: 0	MinQp: 4 MaxQp: 8 Ofs: 0	MinQp: 4 MaxQp: 12 Ofs: 0	MinQp: 0 MaxQp: 4 Ofs: 0	MinQp: 2 MaxQp: 5 Ofs: 0	MinQp: 4 MaxQp: 9 Ofs: 0
rc_range_parameters[2]	MinQp: 1 MaxQp: 5 Ofs: 0	MinQp: 5 MaxQp: 9 Ofs: 0	MinQp: 9 MaxQp: 13 Ofs: 0	MinQp: 1 MaxQp: 5 Ofs: 0	MinQp: 3 MaxQp: 7 Ofs: 0	MinQp: 7 MaxQp: 11 Ofs: 0
rc_range_parameters[3]	MinQp: 1 MaxQp: 6 Ofs: -2	MinQp: 5 MaxQp: 10 Ofs: -2	MinQp: 9 MaxQp: 14 Ofs: -2	MinQp: 1 MaxQp: 6 Ofs: -2	MinQp: 4 MaxQp: 8 Ofs: -2	MinQp: 8 MaxQp: 12 Ofs: -2
rc_range_parameters[4]	MinQp: 3 MaxQp: 7 Ofs: -4	MinQp: 7 MaxQp: 11 Ofs: -4	MinQp: 11 MaxQp: 15 Ofs: -4	MinQp: 3 MaxQp: 7 Ofs: -4	MinQp: 6 MaxQp: 9 Ofs: -4	MinQp: 10 MaxQp: 13 Ofs: -4
rc_range_parameters[5]	MinQp: 3 MaxQp: 7 Ofs: -6	MinQp: 7 MaxQp: 11 Ofs: -6	MinQp: 11 MaxQp: 15 Ofs: -6	MinQp: 3 MaxQp: 7 Ofs: -6	MinQp: 7 MaxQp: 10 Ofs: -6	MinQp: 11 MaxQp: 14 Ofs: -6
rc_range_parameters[6]	MinQp: 3 MaxQp: 7 Ofs: -8	MinQp: 7 MaxQp: 11 Ofs: -8	MinQp: 11 MaxQp: 15 Ofs: -8	MinQp: 3 MaxQp: 7 Ofs: -8	MinQp: 7 MaxQp: 11 Ofs: -8	MinQp: 11 MaxQp: 15 Ofs: -8
rc_range_parameters[7]	MinQp: 3 MaxQp: 8 Ofs: -8	MinQp: 7 MaxQp: 12 Ofs: -8	MinQp: 11 MaxQp: 16 Ofs: -8	MinQp: 3 MaxQp: 8 Ofs: -8	MinQp: 7 MaxQp: 12 Ofs: -8	MinQp: 11 MaxQp: 16 Ofs: -8
rc_range_parameters[8]	MinQp: 3 MaxQp: 9 Ofs: -8	MinQp: 7 MaxQp: 13 Ofs: -8	MinQp: 11 MaxQp: 17 Ofs: -8	MinQp: 3 MaxQp: ~~9~~8 Ofs: -8	MinQp: 7 MaxQp: 1~~3~~2 Ofs: -8	MinQp: 11 MaxQp: 1~~7~~6 Ofs: -8

Table E-5: Selected Rate Control-Related Parameter Recommended Values[a] (Continued)

Syntax Element[b]	At 8bpp/ 8bpc	At 8bpp/ 10bpc	At 8bpp/ 12bpc	At 12bpp/ 8bpc	At 12bpp/ 10bpc	At 12bpp/ 12bpc
rc_range_parameters[9]	MinQp: 3 MaxQp: 10 Ofs: -10	MinQp: 7 MaxQp: 14 Ofs: -10	MinQp: 11 MaxQp: 18 Ofs: -10	MinQp: 3 MaxQp: ~~10~~9 Ofs: -10	MinQp: 7 MaxQp: 1~~4~~3 Ofs: -10	MinQp: 11 MaxQp: 1~~8~~7 Ofs: -10
rc_range_parameters[10]	MinQp: 5 MaxQp: 1~~1~~0 Ofs: -10	MinQp: 9 MaxQp: 1~~5~~4 Ofs: -10	MinQp: 13 MaxQp: 1~~9~~8 Ofs: -10	MinQp: 5 MaxQp: ~~11~~9 Ofs: -10	MinQp: 9 MaxQp: 1~~5~~3 Ofs: -10	MinQp: 13 MaxQp: 1~~9~~7 Ofs: -10
rc_range_parameters[11]	MinQp: 5 MaxQp: ~~12~~11 Ofs: -12	MinQp: 9 MaxQp: 1~~6~~5 Ofs: -12	MinQp: 13 MaxQp: ~~20~~19 Ofs: -12	MinQp: 5 MaxQp: ~~12~~9 Ofs: -12	MinQp: 9 MaxQp: 1~~6~~3 Ofs: -12	MinQp: 13 MaxQp: ~~20~~17 Ofs: -12
rc_range_parameters[12]	MinQp: 5 MaxQp: ~~13~~11 Ofs: -12	MinQp: 9 MaxQp: 1~~7~~5 Ofs: -12	MinQp: 13 MaxQp: ~~21~~19 Ofs: -12	MinQp: 5 MaxQp: ~~13~~9 Ofs: -12	MinQp: 9 MaxQp: 1~~7~~3 Ofs: -12	MinQp: 13 MaxQp: ~~21~~17 Ofs: -12
rc_range_parameters[13]	MinQp: ~~7~~9 MaxQp: 1~~3~~2 Ofs: -12	MinQp: 1~~1~~3 MaxQp: 1~~7~~6 Ofs: -12	MinQp: 1~~5~~7 MaxQp: 2~~1~~0 Ofs: -12	MinQp: 7 MaxQp: 1~~3~~0 Ofs: -12	MinQp: 11 MaxQp: 1~~7~~4 Ofs: -12	MinQp: 15 MaxQp: ~~21~~18 Ofs: -12
rc_range_parameters[14]	MinQp: 1~~2~~2 MaxQp: 1~~5~~3 Ofs: -12	MinQp: 1~~7~~6 MaxQp: 1~~9~~7 Ofs: -12	MinQp: 2~~1~~0 MaxQp: 2~~3~~1 Ofs: -12	MinQp: 1~~3~~0 MaxQp: 1~~5~~1 Ofs: -12	MinQp: 1~~7~~4 MaxQp: 1~~9~~5 Ofs: -12	MinQp: ~~21~~18 MaxQp: ~~23~~19 Ofs: -12

Background Information

Testing has indicated that there are some worst-case clips with motion where some “scintillation” can be seen when the images are coded using the guidance in the DSC 1.1 standard. This SCR intends to address this issue by providing recommended PPS values that improve perceptual quality by making slices taller and allocating more bit budget to the first lines of slices.

- End of Document -