



# Information technology — Generic coding of moving pictures and associated audio information —

## Part 7: Advanced Audio Coding (AAC)

### TECHNICAL CORRIGENDUM 1

*Technologies de l'information — Codage générique des images animées et du son associé —*

*Partie 7: Codage du son avancé (AAC)*

*RECTIFICATIF TECHNIQUE 1*

Technical Corrigendum 1 to International Standard ISO/IEC 13818-7:1997 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

1) Add the following paragraph at the end of clause 5:

“

The number of bits for each data element is written in the second column. "X..Y" indicates that the number of bits is one of the values between X and Y including X and Y. "{X;Y}" means the number of bits is X or Y, depending on the value of other data elements in the bitstream.

”

2) Replace Table 6.13 in subclause 6.3 with the following:

“

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No. of bits           | Mnemonic                                  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|-------------------------------------------|
| <pre> section_data() {     if( window_sequence == EIGHT_SHORT_SEQUENCE )         sect_esc_val = (1&lt;&lt;3) - 1     else         sect_esc_val = (1&lt;&lt;5) - 1      for( g=0; g &lt; num_window_groups; g++ ) {         k=0         i=0         while (k&lt;max_sfb) {             <b>sect_cb[g][i]</b>             sect_len=0             while (<b>sect_len_incr</b> == sect_esc_val)                 sect_len += sect_esc_val             sect_len += sect_len_incr             sect_start[g][i] = k             sect_end[g][i] = k+sect_len             for (sfb=k; sfb&lt;k+sect_len; sfb++)                 sfb_cb[g][sfb] = sect_cb[g][i];             k += sect_len             i++         }         num_sec[g] = i     } } </pre> | <p>4</p> <p>{3;5}</p> | <p><b>uimsbf</b></p> <p><b>uimsbf</b></p> |

”

3) In Table 6.15 in subclause 6.3, replace:

No. of bits “4/6” with “{4;6}”

and

No. of bits “3/5” with “{3;5}”.

4) Replace Table 6.16 in subclause 6.3 with the following:

“

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | No. of bits                                       | Mnemonic                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|---------------------------------------------------------------------------------|
| <pre> spectral_data() {     for( g=0; g&lt;num_window_groups; g++ ) {         for (i=0; i&lt;num_sec[g]; i++) {             if (sect_cb[g][i] != ZERO_HCB &amp;&amp;                 sect_cb[g][i] &lt;= ESC_HCB) {                 for (k=sect_sfb_offset[g][sect_start[g][i];                     k&lt; sect_sfb_offset[g][sect_end[g][i]]; ) {                     if (sect_cb[g][i]&lt;FIRST_PAIR_HCB) {                         <b>hcod</b>[sect_cb[g][i]][w][x][y][z]                         if( unsigned_cb[sect_cb[g][i]] )                             <b>quad_sign_bits</b>                         k += QUAD_LEN                     }                     else {                         <b>hcod</b>[sect_cb[g][i]][y][z]                         if( unsigned_cb[sect_cb[g][i]] )                             <b>pair_sign_bits</b>                         k += PAIR_LEN                     }                 }             }         }     } } </pre> | <p>1..16</p> <p>0..4</p> <p>1..15</p> <p>0..2</p> | <p><b>bslbf</b></p> <p><b>bslbf</b></p> <p><b>bslbf</b></p> <p><b>bslbf</b></p> |

|                               |             |
|-------------------------------|-------------|
| if (sect_cb[g][i]==ESC_HCB) { |             |
| if (y==ESC_FLAG)              |             |
| <b>hcod_esc_y</b>             | 5..21 bslbf |
| if (z==ESC_FLAG)              |             |
| <b>hcod_esc_z</b>             | 5..21 bslbf |
| }                             |             |
| }                             |             |
| }                             |             |
| }                             |             |
| }                             |             |
| }                             |             |
| }                             |             |

5) Replace Table 6.18 in subclause 6.3 with the following:

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | No. of bits                                                                                                                        | Mnemonic                                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| coupling_channel_element()<br>{<br><b>element_instance_tag</b><br><b>ind_sw_cce_flag</b><br><b>num_coupled_elements</b><br>num_gain_element_lists = 0<br>for (c=0; c<num_coupled_elements+1; c++) {<br>num_gain_element_lists++<br><b>cc_target_is_cpe[c]</b><br><b>cc_target_tag_select[c]</b><br>if ( cc_target_is_cpe[c] ) {<br><b>cc_l[c]</b><br><b>cc_r[c]</b><br>if (cc_l[c] && cc_r[c] )<br>num_gain_element_lists++<br>}<br>}<br><b>cc_domain</b><br><b>gain_element_sign</b><br><b>gain_element_scale</b><br><br>individual_channel_stream(0)<br><br>for ( c=1; c<num_gain_element_lists; c++ ) {<br>if ( ind_sw_cce_flag ) {<br>cge = 1<br>} else {<br><b>common_gain_element_present[c]</b><br>cge = common_gain_element_present[c]<br>}<br>if ( cge )<br><b>hcod_sf[common_gain_element[c]]</b><br>else {<br>for (g=0; g<num_window_groups; g++) {<br>for (sfb=0; sfb<max_sfb; sfb++) {<br>if ( sfb_cb[g][sfb] != ZERO_HCB )<br><b>hcod_sf[dpcm_gain_element[c][g][sfb]]</b><br>}<br>}<br>}<br>}<br>} | 4<br>1<br>3<br><br><br>1<br>4<br><br>1<br>1<br>1<br><br><br>1<br>1<br>2<br><br><br><br><br><br><br>1<br><br><br>1..19<br><br>1..19 | <b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b><br><br><br><b>uimsbf</b><br><b>uimsbf</b><br><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b><br><br><br><b>uimsbf</b><br><br><br><b>uimsbf</b><br><br><br><b>bslbf</b><br><br><b>bslbf</b> |

6) Replace Table 6.22 in subclause 6.3 with the following:

“

| Syntax                                                                                                                                                      | No. of bits        | Mnemonic                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|----------------------------------------|
| fill_element()<br>{<br>cnt = <b>count</b><br>if (cnt == 15)<br>cnt += <b>esc_count</b> - 1;<br>while (cnt > 0) {<br>cnt -= extension_payload(cnt)<br>}<br>} | 4<br><br><br><br>8 | <b>uimsbf</b><br><br><br><b>uimsbf</b> |

”

and add the following tables, Table 6.24, Table 6.25 and Table 6.26, at the end of subclause 6.3:

“

**Table 6.24 — Syntax of extension\_payload()**

|                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                      |                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| extension_payload(cnt)<br>{<br><b>extension_type</b><br>switch( extension_type ) {<br>case EXT_DYNAMIC_RANGE:<br>n = dynamic_range_info();<br>return n;<br>case EXT_FILL_DATA:<br><b>fill_nibble</b> /* must be '0000' */<br>for (i=0; i<cnt-1; i++)<br><b>fill_byte[i]</b> /* must be '10100101' */<br>return cnt<br>case default:<br>for (i=0; i<8*(cnt-1)+4; i++)<br><b>other_bits[i]</b><br>return cnt<br>}<br>} | 4<br><br><br><br><br><br>4<br><br>8<br><br><br><br>1 | <b>uimsbf</b><br><br><br><br><br><br><b>uimsbf</b><br><br><b>uimsbf</b><br><br><br><br><b>uimsbf</b> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|------------------------------------------------------------------------------------------------------|

**Table 6.25 — Syntax of dynamic\_range\_info()**

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                           | No. of bits                                                             | Mnemonic                                                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| dynamic_range_info()<br>{<br>n = 1<br>drc_num_bands = 1<br><b>pce_tag_present</b><br>if (pce_tag_present == 1) {<br><b>pce_instance_tag</b><br><b>drc_tag_reserved_bits</b><br>n++<br>}<br><b>excluded_chns_present</b><br>if (excluded_chns_present == 1) {<br>n += excluded_channels()<br>}<br><b>drc_bands_present</b><br>if (drc_bands_present == 1) {<br><b>drc_band_incr</b><br><b>drc_bands_reserved_bits</b><br>n++<br>} | <br><br><br>1<br><br><br>4<br>4<br><br><br>1<br><br><br>1<br><br>4<br>4 | <br><br><br><b>uimsbf</b><br><br><br><b>uimsbf</b><br><b>uimsbf</b><br><br><br><b>uimsbf</b><br><br><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b> |

|                                                                                                                                          |             |                            |
|------------------------------------------------------------------------------------------------------------------------------------------|-------------|----------------------------|
| drc_num_bands = drc_num_bands + drc_band_incr<br>for (i=0; i<drc_num_bands; i++) {<br><b>drc_band_top[i]</b><br>n++<br>}                 | 8           | uimsbf                     |
| prog_ref_level_present<br>if (prog_ref_level_present == 1) {<br><b>prog_ref_level</b><br><b>prog_ref_level_reserved_bits</b><br>n++<br>} | 1<br>7<br>1 | uimsbf<br>uimsbf<br>uimsbf |
| for (i=0; i<drc_num_bands; i++) {<br><b>dyn_rng_sgn[i]</b><br><b>dyn_rng_ctl[i]</b><br>n++<br>}                                          | 1<br>7      | uimsbf<br>uimsbf           |
| return n<br>}                                                                                                                            |             |                            |

Table 6.26 — Syntax of excluded\_channels()

| Syntax                                                                                                                                                                                                                                                                                                      | No. Of bits | Mnemonic                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|----------------------------|
| excluded_channels( )<br>{<br>n = 0<br>num_excl_chan = 7<br>for (i=0; i<7; i++)<br><b>exclude_mask[ i ]</b><br>n++<br>while ( <b>additional_excluded_chns[n-1]</b> == 1) {<br>for (i= num_excl_chan; i< num_excl_chan+7; i++)<br><b>exclude_mask[ i ]</b><br>n++<br>num_excl_chan += 7<br>}<br>return n<br>} | 1<br>1<br>1 | uimsbf<br>uimsbf<br>uimsbf |

”

7) In subclause 7.1, replace “ATDS0” with “ADTS”.

8) Replace definition of **num\_coupled\_channels** in subclause 7.3.2 with the following:

“  
**num\_coupled\_elements**                      number of coupled target elements  
 ”

9) Replace definition of **home** in subclause 8.1.1 with the following:

“  
 see ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.2) definition for **original\_copy**  
 ”

10) Replace definition of **original\_copy** in subclause 8.1.1 with the following:

“  
 see ISO/IEC 11172-3, subclause 2.4.2.3 (table 6.2) definition for **copyright**  
 ”

11) Replace the last sentence in the second paragraph of subclause 8.1.2 with the following:

“

However, one non-normative transport stream, called Audio\_Data\_Transport\_Stream (ADTS), is described. It may be used for applications in which the decoder can parse this stream.

”

12) Replace the first two paragraphs of subclause 8.2.3 with the following:

“

Assuming that the start of a raw\_data\_block is known, it can be decoded without any additional «transport-level» information and produces 1024 audio samples per output channel. The sampling rate of the audio signal, as specified by the **sampling\_frequency\_index**, may be specified in a program\_config\_element or it may be implied in the specific application domain. In the latter case, the **sampling\_frequency\_index** must be deduced in order for the bitstream to be parsed. Since a given **sampling\_frequency\_index** is associated with only one sampling frequency, and since maximum flexibility is desired in the range of possible sampling frequencies, the following table shall be used to associate an implied sampling frequency with the desired **sampling\_frequency\_index**. It is used as follows: identify the frequency in the table that is the highest frequency that is less than or equal to the implied frequency, and use the index that is in that same row.

| Frequency | sampling_frequency_index |
|-----------|--------------------------|
| 92017     | 0x0                      |
| 75132     | 0x1                      |
| 55426     | 0x2                      |
| 46009     | 0x3                      |
| 37566     | 0x4                      |
| 27713     | 0x5                      |
| 23004     | 0x6                      |
| 18783     | 0x7                      |
| 13856     | 0x8                      |
| 11502     | 0x9                      |
| 9391      | 0xa                      |
| 0         | 0xb                      |

Assuming that the start of the first raw\_data\_block in a raw\_data\_stream is known, the sequence can be decoded without any additional “transport-level” information and produces 1024 audio samples per raw\_data\_block per output channel.

”

13) Replace the second item of the second level bulleted list in subclause 8.3.5 with the following:

“

If there is only one group with length eight (num\_window\_group = 1, window\_group\_length[0]=8), the results is that spectral data of all eight SHORT\_WINDOWs is interleaved by scalefactor window bands.

”

14) Replace definition of **num\_valid\_cce\_elements** in subclause 8.5 with the following:

“

**num\_valid\_cc\_elements**      The number of CCE's that can add to the audio data for this program (Table 6.21)

”

15) Replace definition of **valid\_cce\_element\_tag\_select** in subclause 8. 5 with the following:

“

**valid\_cc\_element\_tag\_select**   instance\_tag of the CCE addressed (Table 6.21)

”

16) Replace subclause 8.7 with the following:

“

## 8.7 Fill element (FIL) including Dynamic Range Control (DRC)

Bitstream elements:

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>count</b>                        | Initial value for length of fill data (Table 6.22)                                                                                                                                                                                                                                                                                                                                                                            |
| <b>esc_count</b>                    | Incremental value of length of fill data (Table 6.22)                                                                                                                                                                                                                                                                                                                                                                         |
| <b>extension_type</b>               | Four bit field indicating the type of fill element content (Table 6.22)                                                                                                                                                                                                                                                                                                                                                       |
| <b>fill_nibble</b>                  | Four bit field for fill (Table 6.24)                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>fill_byte</b>                    | Byte to be discarded by the decoder (Table 6.24)                                                                                                                                                                                                                                                                                                                                                                              |
| <b>other_bits</b>                   | Bits to be discarded by the decoder (Table 6.24)                                                                                                                                                                                                                                                                                                                                                                              |
| <b>pce_tag_present</b>              | One bit indicating that program element tag is present (table 6.25).                                                                                                                                                                                                                                                                                                                                                          |
| <b>pce_instance_tag</b>             | Tag field that indicates with which program the dynamic range information is associated (table 6.25)                                                                                                                                                                                                                                                                                                                          |
| <b>drc_tag_reserved_bits</b>        | Reserved (table 6.25)                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>excluded_chns_present</b>        | One bit indicating that excluded channels are present (table 6.25)                                                                                                                                                                                                                                                                                                                                                            |
| <b>drc_bands_present</b>            | One bit indicating that DRC multi-band information is present (table 6.25)                                                                                                                                                                                                                                                                                                                                                    |
| <b>drc_band_incr</b>                | Number of DRC bands greater than 1 having DRC information (table 6.25)                                                                                                                                                                                                                                                                                                                                                        |
| <b>drc_bands_reserved_bits</b>      | Reserved (table 6.25)                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>drc_band_top[i]</b>              | Indicates top of i-th DRC band in units of 4 spectral lines (table 6.25).<br>If $drc\_band\_top[i]=k$ , then the index (w.r.t zero) of the highest spectral coefficient that is in the i-th DRC band is $= k*4+3$ . In case of an EIGHT_SHORT_SEQUENCE window_sequence the index is interpreted as pointing into the concatenated array of $8*128$ (de-interleaved) frequency points corresponding to the 8 short transforms. |
| <b>prog_ref_level_present</b>       | One bit indicating that reference level is present (table 6.25).                                                                                                                                                                                                                                                                                                                                                              |
| <b>prog_ref_level</b>               | Reference level. A measure of long-term program audio level for all channels combined (table 6.25).                                                                                                                                                                                                                                                                                                                           |
| <b>prog_ref_level_reserved_bits</b> | Reserved (table 6.25)                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>dyn_rng_sgn[i]</b>               | Dynamic range control sign information. One bit indicating the sign of $dyn\_rng\_ctl$ (0 if positive, 1 if negative, table 6.25)                                                                                                                                                                                                                                                                                             |
| <b>dyn_rng_ctl[i]</b>               | Dynamic range control magnitude information (table 6.25)                                                                                                                                                                                                                                                                                                                                                                      |
| <b>exclude_mask[i]</b>              | Boolean array indicating the audio channels of a program that are excluded from DRC processing using this DRC information.                                                                                                                                                                                                                                                                                                    |
| <b>additional_excluded_chns[i]</b>  | One bit indicating that additional excluded channels are present (table 6.26)                                                                                                                                                                                                                                                                                                                                                 |

Fill elements have to be added to the bitstream if the total bits for all audio data together with all additional data is lower than the minimum allowed number of bits in this frame necessary to reach the target bitrate. Dynamic Range Control (DRC) bits must be added to the fill element whenever the encoder wishes to include DRC information. Under normal conditions fill bits are avoided and free bits are used to fill up the bit reservoir. Fill bits are written only if the bit reservoir is full. Any number of fill elements are allowed.

### Decoding process:

The syntactic element **count** gives the initial value of the length of the fill data. In the same way as for the data element this value is incremented with the value of **esc\_count** if **count** equals 15. The resulting number gives the number of bytes to be read.

### DRC Decoding process:

Fill elements containing an extension\_payload with a extension\_type of EXT\_DYNAMIC\_RANGE (see below) are reserved for dynamic range information. In this case the fill\_element **count** field must be set equal to the total length, in bytes, of all dynamic range information plus the extension\_type field.

**prog\_ref\_level\_present** indicates that **prog\_ref\_level** is being transmitted. This permits **prog\_ref\_level** to be sent as infrequently as desired (e.g. once), although periodic transmission would permit break-in.

**prog\_ref\_level** is quantized in 0.25 dB steps using 7 bits, and therefore has a range of approximately 32 dB. It indicates program level relative to full scale (i.e. dB below full scale), and is reconstructed as:

$$level = 32767 \cdot 2^{-prog\_ref\_level/24}$$