Telecommunication Standardization Sector
Study Group 15
Experts Group for ATM Video Coding
(Rapporteur's Group on Part of Q.2/15)

Document AVC-483
March 1993

INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO-IEC/JTC1/SC29/WG11
CODED REPRESENTATION OF PICTURE AND AUDIO INFORMATION

ISO-IEC/JTC1/SC29/WG11/MPEG92/321

Source: M. E. Nilsson, I. Parke, BT Labs
Title: Sequence layer syntax
Purpose: Discussion and proposal

This document discusses some of the limitations of the current sequence level syntax and proposes changes to overcome these limitations. All proposed changes are made in a way that retains syntax compatibility with MPEG-1.

1. MPEG-1 syntax compatibility

In order to maximise the opportunity for the interworking of MPEG-1 and MPEG-2 systems, it is essential that the MPEG-2 syntax is defined to be compatible with the MPEG-1 syntax. This implies that a bitstream that has been generated by a MPEG-1 encoder, without knowledge of the decoder system capabilities, will always satisfy the MPEG-2 syntax.

2. The purpose of the sequence header

The sequence header contains information that is required by a decoder when it first starts to decode a bitstream. It must therefore occur at the beginning of the bitstream.

The sequence header will be needed at later points in the coded bitstream for some applications to:

- provide random access capability;
- change coding parameters, such as, quantization matrices.

At present when sequence data is repeated, it is only possible to change the quantization matrices. It is therefore not possible at present to change the prediction weighting matrices within a coded sequence. It is proposed this is changed to allow these weights to be changed.

There is a complication for an encoder that wants to change such coding parameters, as the current syntax requires the transmission of a complete sequence header, then a group of pictures header, and then an intra picture. This represents quite a serious overhead, especially for applications with no interest in coding intra pictures.

This problem can be solved by providing another mechanism in addition to or instead of the sequence header for changing these coding parameters. Another solution is to remove the restriction that sequence headers must be followed by group of pictures headers and intra pictures; this is discussed in more detail in the next section of this document.

3. Start codes and syntax layers

As outlined in the above section, the current syntax is awkward in the way it handles start codes. For example, user data can follow extension data but extension data can not come after user data. This is considered to be due to the method used to describe the syntax rather any particular desire to prevent encoders generating extension data after generating user data. This example is insignificant, but what is more important is that at present, if an encoder desires to encode sequence header data, it must follow this with a group of pictures header and then very significantly, an intra picture. This has serious implications to an encoder that is not encoding intra pictures but wants to change parameters in the sequence header, such as the quantization weighting matrix or the prediction weights.

It is proposed that the syntax is made more flexible to allow user data to appear before extension data, and more importantly, for sequence header data to occur without the need to have proceeding group of pictures and intra picture data.

It is also suggested that the group of pictures layer is not necessary for all applications, and for some is a major inconvenience as it demands the coding of the next picture as an intra picture. The group of pictures layer was invented to help with random access and editing. The start of a group of pictures is a good point at which to start decoding, as an intra picture is present and closed_gop and broken_link indicate whether the interpolated pictures temporally before the intra can be decoded correctly. This is definitely useful information to be able to put into the coded bitstream, but why should it have the status of being a syntax 'layer'? It is proposed here that the status of the group of pictures header is changed from being a syntax 'layer' to being a syntax element in the sequence layer. Note, it would still be possible, even desirable, for the semantics to state that the next picture after group of pictures data, must be an intra picture.

This proposal is extended to the sequence header, so that the sequence header, renamed to the more appropriate 'sequence_data', becomes a syntax element at the sequence layer rather than being the header for the sequence layer.

This proposal is then completed by giving the same status to sequence_data, group_of_pictures_data, pictures, extension_data and user_data. The syntax is illustrated in the figure 1. As this type of diagram illustrates syntax clearly, it would be beneficial to include this type of diagram in the Working Draft for all the syntax layers.

4. Sequence layer syntax proposal

Figure 1 shows the proposal for the sequence layer syntax. This shows that sequence header data can be present in the bitstream without the need to be followed by group of pictures data.

This syntax is compatible with MPEG-1 because the MPEG-1 syntax is a valid form of this syntax. Consequently, an MPEG-2 decoder that can decode this syntax will be able to decode any MPEG-1 bitstream.

Although this new syntax can be expressed with current MPEG notation, it would be awkward and difficult to read. An alternative is shown in table 1 below. This can be seen to mirror the structure of the diagram of figure 1. It is consequently easy to read. It takes advantage of a new command, named here as 'choose'. It is based on the concept of the OCCAM command ALT (meaning alternative); the meaning is that one of the parameter functions of the command are processed. In the MPEG case, this means that the encoder can choose to insert into the syntax at the point any one of the arguments of the choose command. The decoder must therefore be designed to cope with any of these alternatives. Although this may at first seem complicated, it is not, and is exactly the same requirement as that made by the command 'nextbits'. This syntax can be defined using the nextbits function, but this new method of expressing the same syntax is much clearer.

5. Profiles and capabilities

At present, the sequence header gives only a little indication of what is to follow in the bitstream. In the MPEG-1 syntax, the constrained parameters flag allowed a decoder to judge whether it could decode a bitstream without having to try it until it failed. A similar indication would be beneficial to the MPEG-2 syntax.

It is most likely that this will require more than a single bit flag, due to the many profiles being considered. However, a few bits would be sufficient to indicate the profile and the profile level used.

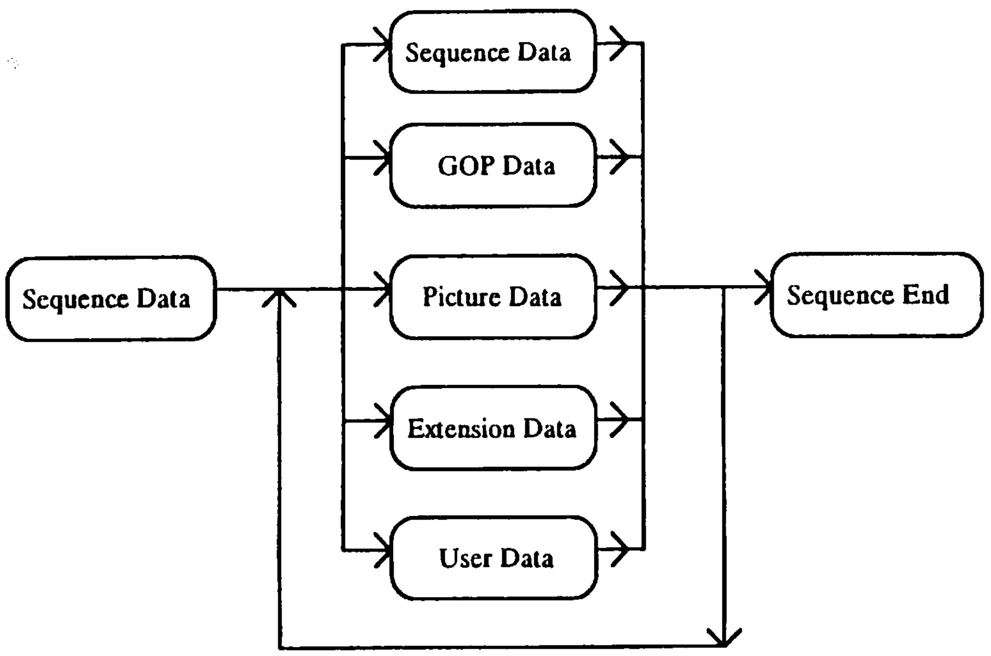

Figure 1. Proposal for sequence layer syntax.

This clear indication of the character of the bitstream can also lead to a simplification of the parameters of the sequence header. The various sequence level additions to the syntax could be arranged into logical groups depending on the functionality they provide, that is, the profile with which they are associated.

Table 2 shows the proposed syntax for the sequence header, renamed for previously described reasons to sequence_data. This consists of the MPEG-1 sequence header with the conditional attachment of sequence_data_extension.

Table 3 shows the proposed syntax for the sequence_data_extension. After the sequence extension start code, there is a sequence extension code identifier. It is suggested that this is used in a similar way to start code identifiers used with ordinary start codes. It would be a byte indicating the type of extension data that follows. This is considered to be preferable to the method of identifying the meaning of extension data purely on the basis of its position in the bitstream; it should be remembered that future extensions to the bitstream may be made.

The extension start code identifier is followed by a profile indicator. It is suggested that this would indicate the profile and level of profile present. It would indicate whether progressive or interlaced pictures are to be coded, whether spatial or frequency scalability is to be used, etc. It could also indicate the profile level, that is, picture resolution and rate, bit rate, motion vector ranges, etc.; two or three bits should be sufficient for this, perhaps indicating capabilities of the order of QCIF, SIF, TV, HDTV, > > HDTV.

The profile indicator is followed by extension_sequence_data_flags. This is an array of flags which would indicate which 'extended' sequence data syntax elements are present in the bitstream. At shown in table 3, there would be three flags indicating the presence of spatially embedded sequence data, fscalable sequence data and general extended sequence data. The syntax for these data are shown in tables 4,5, and 6.

Table 7 shows the group of pictures data syntax under the new sequence layer proposal. It is immediately obvious this is much cleaner than the existing syntax for group [illegible] data, as the need to consider extension and user data following the group data has been removed.

It may be beneficial to provide a mechanism for coding the time code for applications that do not code intra pictures and consequently do not code group of pictures data. This could be done using the extension data, shown in table 8, with the extension start code identifier indicating that the extension data was a time code.

6. Conclusion

This document discussed some of the limitations of the current sequence level syntax and proposed changes to overcome these limitations. All proposed changes were made in a way that retained syntax compatibility with MPEG-1.

```
video_sequence() {
  next_start_code()
  sequence_data()
  do {
    choose {
      sequence_data()
      group_of_pictures_data()
      picture()
      extension_data()
      user_data()
      }
  } while ( nextbits() != sequence_end_code )
  sequence_end_code                                    32        bslbf
}
```

Table 1. Sequence layer syntax proposal

sequence_data_extension_code is 000001B5xx in hexadecimal.

```
sequence_data() {
  sequence_header_code                                 32        bslbf
  horizontal_size_value                                12        uimsbf
  vertical_size_value                                  12        uimsbf
  pel_aspect_ratio                                     4         uimsbf
  picture_rate                                         4         uimsbf
  bit_rate                                             18        uimsbf
  marker_bit                                           1         "1"
  vbv_buffer_size                                      10        uimsbf
  constrained_parameter_flag                           1
  load_intra_quantizer_matrix                          1
  if ( load_intra_quantizer_matrix )
     intra_quantizer_matrix[64]                        8*64      uimsbf
  load_non_intra_quantizer_matrix                      1
  if ( load_non_intra_quantizer_matrix )
     non_intra_quantizer_matrix[64]                    8*64      uimsbf
  next_start_code()
  if (nextbits() = = sequence_data_extension_code)
    sequence_data_extension()
}
```

Table 2. Sequence data syntax proposal

sequence_data_extension() {		
extension_start_code	32	bslbf
extension_start_code_identifier	8	uimsbf
profile_indication	8	uimsbf
extended_sequence_data_flags	3	uimsbf
if (spatial_embedded) { spatial_embedded_sequence_data() }		
if (fscalable) { fscalable_sequence_data() }		
if (general_extended) { general_extended_sequence_data() }		
next_start_code()		
}		

Table 3. Sequence data extension syntax proposal

spatial_embedded_sequence_data() {		
compatible_mtype	2	uimsbf
load_prediction_weighting_matrix	1	uimsbf
if (load_prediction_weighting_matrix)		
prediction_weighting_matrix[8]	4*8	uimsbf
low_resolution_prediction_horizontal_dimension	17	uimsbf
low_resolution_prediction_vertical_dimension	17	uimsbf
}		

Table 4. Spatial embedded sequence data syntax proposal

fscalable_sequence_data() {		
interlaced	1	uimsbf
subband	1	uimsbf
linked_prediction	1	uimsbf
scalable_side_information	1	uimsbf
motion_refinement	1	uimsbf
MUVLC_coding	1	uimsbf
do {		
fscale_code	8	uimsbf
motion_compensation_loop	1	uimsbf
} while (nextbits != '00000111')		
end_of_fscales_code	8	'00000111'
low_resolution_prediction_horizontal_dimension	17	uimsbf
low_resolution_prediction_vertical_dimension	17	uimsbf
}		

Table 5. Fscalable sequence layer syntax proposal

general_extended_sequence_data() {		
chroma_format	2	uimsbf
horizontal_size_extension	4	uimsbf
vertical_size_extension	4	uimsbf
video_format	3	uimsbf
intra_vlc_format	1	uimsbf
tcoef_escape_format	1	uimsbf
display_horizontal_dimension	16	uimsbf
display_vertical_dimension	16	uimsbf
10-bit_input_flag	1	uimsbf
}		

Table 6. General extended sequence layer syntax proposal

group_of_pictures_data() {		
group_start_code	**32**	**bslbf**
time_code	**25**	
closed_gop	**1**	
broken_link	**1**	
next_start_code()		
}		

Table 7. Simplified group of pictures data syntax proposal

extension_data() {		
extension_start_code	**32**	**bslbf**
extension_start_code_identifier	**8**	**uimsbf**
if (nextbits() != start_code)		
extension_data	**8**	**uimsbf**
}		

Table 8. Extension data syntax proposal

user_data() {		
user_start_code	**32**	**bslbf**
if (nextbits() != start_code)		
user_data	**8**	**uimsbf**
}		

Table 9. User data syntax proposal