INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.263

Annex W

(11/2000)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving video

---

Video coding for low bit rate communication

Annex W: Additional supplemental enhancement information specification

ITU-T Recommendation H.263 – Annex W

(Formerly CCITT Recommendation)

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	**H.260–H.279**
Related systems aspects	H.280–H.299
SYSTEMS AND TERMINAL EQUIPMENT FOR AUDIOVISUAL SERVICES	H.300–H.399
SUPPLEMENTARY SERVICES FOR MULTIMEDIA	H.450–H.499

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation H.263

Video coding for low bit rate communication

ANNEX W

Additional supplemental enhancement information specification

Summary

Optional *Additional Supplemental Enhancement Information* which can be added to an H.263 bitstream to provide backward-compatible enhancements, including:

- Indication of use of a specific fixed-point IDCT.
- Picture Messages, including the message types of:
  - Arbitrary Binary Data;
  - Text (Arbitrary, Copyright, Caption, Video Description, or Uniform Resource Identifier);
  - Picture Header Repetition (Current, Previous, Next with Reliable Temporal Reference, or Next with Unreliable Temporal Reference);
  - Interlaced Field Indications (Top or Bottom); and
  - Spare Reference Picture Identification.

Source

Annex W to ITU-T Recommendation H.263 was prepared by ITU-T Study Group 16 (2001-2004) and approved under the WTSA Resolution 1 procedure on 17 November 2000.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

ITU-T Recommendation H.263

Video coding for low bit rate communication

ANNEX W

Additional supplemental enhancement information specification

W.1 Scope

This annex describes the format of the additional supplemental enhancement information sent in the PSUPP field of the picture layer of H.263, which adds to the functionality defined in Annex L. The capability of a decoder to provide any or all of the capabilities described in this annex may be signalled by external means (for example, ITU-T H.245). Decoders which do not provide the additional capabilities may simply discard any of the newly defined PSUPP information bits that appear in the bitstream. The presence of this supplemental enhancement information is indicated by the presence of both the PEI bit, and by the following PSUPP octet whose FTYPE field has one of the two newly defined values. The basic interpretation of PEI, PSUPP, FTYPE, and DSIZE is identical to Annex L and to clauses 5.1.24 and 5.1.25.

W.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below.

– ISO/IEC 10646-1:2000: *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*.

– IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.

W.3 Additional FTYPE Values

Two values that were reserved in Annex L, Table L.1 are defined as in Table W.1.

Table W.1/H.263 – FTYPE function type values

13	Fixed-Point IDCT
14	Picture Message

W.4 Recommended maximum number of PSUPP octets

When using any of the aforementioned FTYPE functions defined in this annex, the total number of PSUPP octets per picture should, in relation to the coded picture size, be kept reasonably small, and should not exceed 256 octets regardless of the coded picture size.

NOTE – Some data transmission protocols used for conveyance of the video bitstream may provide for external repetition of picture header contents for error resilience purposes, and may place limits on the amount of such data that can be repeated from a picture header (e.g. 504 bits in the IETF RFC 2429 packetization format). The inclusion of a large number of PSUPP octets may result in the lack of such an external protocol to provide for full repetition of the picture header contents.

W.5 Fixed-point IDCT

The fixed-point IDCT function indicates that a particular IDCT approximation is used in construction of the bitstream. DSIZE shall be equal to 1 for the fixed-point IDCT function. The octet of PSUPP data that follows specifies the particular IDCT implementation. A value of 0 indicates the reference IDCT 0 as described in W.5.3; values of 1 through 255 are reserved.

W.5.1 Decoder operation

The capability of a decoder to perform a particular fixed-point IDCT may be signalled to the encoder by external means (for example, ITU-T H.245). When receiving an encoded bitstream with the fixed-point IDCT indication, a decoder shall use the particular fixed-point IDCT if it is capable of doing so.

W.5.2 Removal of forced updating

Annex A specifies the accuracy requirements for the inverse discrete cosine transform (IDCT), allowing numerous compliant implementations. To control accumulation of errors due to mismatched IDCTs at the encoder and decoder, clause 4.4, Forced Updating, requires that macroblocks be coded in INTRA mode at least once every 132 times when coefficients are transmitted.

If the fixed-point IDCT function type is indicated in the bitstream, then the forced updating requirement is removed, and the frequency of INTRA coding is unregulated. An encoder should continue to use forced updating, however, unless it has ascertained through external means that the decoder is capable of the particular fixed-point IDCT specified herein; otherwise there may be mismatch.

W.5.3 Reference IDCT 0

The reference IDCT 0 is any implementation that, for every input block, produces identical output values as the C source program listed below.

NOTE – This fixed-point IDCT is compliant with Annex A/H.263, but is not compliant with the extended range of values requirement in Annex A of ITU-T Rec. H.262 | ISO/IEC 13818-2.

```
/*****************************************************************************
 *
 *                            FIXED-POINT IDCT
 *
 * Fixed-point fast, separable idct
 * Storage precision: 16 bits signed
 * Internal calculation precision: 32 bits signed
 * Input range: 12 bits signed, stored in 16 bits
 * Output range: [-256, +255]
 * All operations are signed
 *
 *****************************************************************************/

/*
 * Includes
 */

#include <stdlib.h>
#include <stdio.h>

/*
 * Typedefs
 */

typedef short int REGISTER;  /* 16 bits signed */
typedef long int LONG; /* 32 bits signed */
```

```
/*
 * Global constants
 */

const REGISTER cpo8   = 0x539f;     /* 32768*cos(pi/8)*1/sqrt(2)    */
const REGISTER spo8   = 0x4546;     /* 32768*sin(pi/8)*sqrt(2)    */
const REGISTER cpo16  = 0x7d8a;     /* 32768*cos(pi/16)   */
const REGISTER spo16  = 0x18f9;     /* 32768*sin(pi/16)   */
const REGISTER c3po16 = 0x6a6e;     /* 32768*cos(3*pi/16) */
const REGISTER s3po16 = 0x471d;     /* 32768*sin(3*pi/16) */
const REGISTER OoR2   = 0x5a82;     /* 32768*1/sqrt(2)    */

/*
 * Function declarations
 */

void Transpose(REGISTER block[64]);
void HalfSwap(REGISTER block[64]);
void Swap(REGISTER block[64]);
void Scale(REGISTER block[64], signed char sh);
void Round(REGISTER block[64], signed char sh,
           const REGISTER min, const REGISTER max);
REGISTER Multiply(const REGISTER a, REGISTER x, signed char sh);
void Rotate(REGISTER *x, REGISTER *y,
            signed char sha, signed char shb,
            const REGISTER a, const REGISTER b,
            int inv);
void Butterfly(REGISTER column[8], char pass);
void IDCT(REGISTER block[64]);

/*
 * Transpose():
 *      Transpose a block
 * Input:
 *      REGISTER block[64]
 * Output:
 *      block
 * Return value:
 *      none
 */
void Transpose(REGISTER block[64])
{
  int i, j;
  REGISTER temp;

  for (i=0; i<8; i++) {
    for (j=0; j<i; j++) {
      temp = block[8*i+j];
      block[8*i+j] = block[8*j+i];
      block[8*j+i] = temp;
    }
  }
  return;
}

/*
 * HalfSwap():
 *      One-dimensional swap
 * Input:
 *      REGISTER block[64]
 * Output:
 *      block
 * Return value:
 *      none
 */
void HalfSwap(REGISTER block[64])
{
  int i;
  REGISTER temp;
```

```
  for (i=0; i<8; i++) {
    temp = block[8+i];
    block[8+i] = block[32+i];
    block[32+i] = temp;
    temp = block[24+i];
    block[24+i] = block[48+i];
    block[48+i] = temp;
    temp = block[40+i];
    block[40+i] = block[56+i];
    block[56+i] = temp;
  }
  return;
}

/*
 * Swap():
 *      Swap and transpose a block
 * Input:
 *      REGISTER block[64]
 * Output:
 *      block
 * Return value:
 *      none
 */
void Swap(REGISTER block[64])
{
  HalfSwap(block);
  Transpose(block);
  HalfSwap(block);
}

/*
 * Scale():
 *      Scale a block
 * Input:
 *      REGISTER block[64]
 *      signed char sh
 * Output:
 *      block
 * Return value:
 *      none
 */
void Scale(REGISTER block[64], signed char sh)
{
  int i;

  if (sh>0) {
    for (i=0; i<64; i++)
      block[i] >>= sh;
  }
  else {
    for (i=0; i<64; i++)
      block[i] <<= -sh;
  }
}

/*
 * Round():
 *      Performs the final rounding of an 8x8 block
 * Input:
 *      REGISTER block[64]
 *      signed char sh
 *      const REGISTER min
 *      const REGISTER max
 * Output:
 *      block
 * Return value:
 *      none
 */
void Round(REGISTER block[64], signed char sh,
           const REGISTER min, const REGISTER max)
```

```
{
  int i;

  for (i=0; i<64; i++) {
    if (block[i] < 0x00007FFF – (1<<(sh-1)))
      block[i] += (1<<(sh-1));
    else
      block[i] = 0x00007FFF;
    block[i] >>= sh;
    block[i] = (block[i]<min) ? min : ((block[i]>max) ? max : block[i]);
  }
  return;
}

/*
 * Multiply():
 *      Multiply by a constant with shift
 * Input:
 *      const REGISTER a
 *      REGISTER x
 *      signed char sh
 * Output:
 *      none
 * Return value:
 *      REGISTER, the result of the multiply
 */
REGISTER Multiply(const REGISTER a, REGISTER x, signed char sh)
{
  LONG tmp;
  REGISTER reg_out;

  /* multiply */
  tmp = (LONG)a * (LONG)x;

  /* shift */
  if (sh > 0)
    tmp >>= sh;
  else
    tmp <<= -sh;

  /* rounding and saturating */
  if (tmp < 0x7FFFFFFF – 0x00007FFF)
    tmp = tmp + 0x00007FFF;
  else
    tmp = 0x7FFFFFFF;

  reg_out = (REGISTER)(tmp >>16);

  return(reg_out);
}

/*
 * Rotate():
 *      Perform rotate operation on two registers
 * Input:
 *      REGISTER *x        pointer to the 1st register
 *      REGISTER *y        pointer to the 2nd register
 *      signed char sha    shift associated with factor a
 *      signed char shb    shift associated with factor b
 *      const REGISTER a   factor a
 *      const REGISTER b   factor b
 *      int inv            1 for inverse dct, 0 for forward dct
 * Output:
 *      *x, *y
 * Return value:
 *      none
 */
void Rotate(REGISTER *x, REGISTER *y,
            signed char sha, signed char shb,
            const REGISTER a, const REGISTER b,
            int inv)
```

```
{
  LONG tmplxa, tmplya, tmplxb, tmplyb;
  LONG tmpl1, tmpl2;

  /*
   * intermediate calculation
   */

  tmplxa = (LONG)(*x) * (LONG)a;
  if (sha > 0)
    tmplxa >>= sha;
  else
    tmplxa <<= -sha;

  tmplya = (LONG)(*y) * (LONG)a;
  if (sha > 0)
    tmplya >>= sha;
  else
    tmplya <<= -sha;

  tmplxb = (LONG)(*x) * (LONG)b;
  if (shb > 0)
    tmplxb >>= shb;
  else
    tmplxb <<= -shb;

  tmplyb = (LONG)(*y) * (LONG)b;
  if (shb > 0)
    tmplyb >>= shb;
  else
    tmplyb <<= -shb;

  /*
   * rounding and rotation
   */

  if (inv) {
    tmplxa += 0x00007FFF;
    tmplxb += 0x00007FFF;

    tmpl1 = tmplxb – tmplya;
    tmpl2 = tmplxa + tmplyb;
  }
  else {
    tmplya += 0x00007FFF;
    tmplyb += 0x00007FFF;

    tmpl1 =  tmplxb + tmplya;
    tmpl2 = -tmplxa + tmplyb;
  }

  /*
   * final rounding
   */

  *x = (REGISTER) (tmpl1 >>16);
  *y = (REGISTER) (tmpl2 >>16);

  return;
}

/*
 * Butterfly():
 *      Perform 1D IDCT on a column
 * Input:
 *      REGISTER column[8]
 *      char pass
```

```
 * Output:
 *      column
 * Return value:
 *      none
 */
void Butterfly(REGISTER column[8], char pass)
{
  int i;
  REGISTER shadow_column[8];

  /*
   * For readability, we use a shadow column
   * that contains the state of column at the
   * preceding stage of the butterfly.
   */

  /*
   * Initialization
   */

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * First Phase
   */

  Rotate(column+2, column+6, pass-2, pass-1, cpo8,   spo8,   1);
  Rotate(column+1, column+7, pass-1, pass-1, cpo16,  spo16,  1);
  Rotate(column+3, column+5, pass-1, pass-1, c3po16, s3po16, 1);

  if (pass) {
    int a, tmp=column[4], b=column[0];
    a = b+tmp;
    b = b-tmp;
    column[0] = (a – ((tmp<0) ? 1 : 0)) >> 1;
    column[4] = (b – ((tmp<0) ? 1 : 0)) >> 1;
  }
  else {
    column[0] = shadow_column[0] + shadow_column[4];
    column[4] = shadow_column[0] – shadow_column[4];
  }

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * Second Phase
   */

  column[1] = shadow_column[1] – shadow_column[3];
  column[3] = shadow_column[1] + shadow_column[3];

  column[7] = shadow_column[7] – shadow_column[5];
  column[5] = shadow_column[7] + shadow_column[5];

  column[0] = shadow_column[0] + shadow_column[6];
  column[6] = shadow_column[0] – shadow_column[6];

  column[4] = shadow_column[4] + shadow_column[2];
  column[2] = shadow_column[4] – shadow_column[2];

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * Third Phase
   */

  column[7] = shadow_column[7] – shadow_column[3];
  column[3] = shadow_column[7] + shadow_column[3];
```

```
  column[1] = Multiply(OoR2, shadow_column[1], -2);
  column[5] = Multiply(OoR2, shadow_column[5], -2);

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * Fourth Phase
   */

  column[4] = shadow_column[4] + shadow_column[3];
  column[3] = shadow_column[4] – shadow_column[3];

  column[2] = shadow_column[2] + shadow_column[7];
  column[7] = shadow_column[2] – shadow_column[7];

  column[0] = shadow_column[0] + shadow_column[5];
  column[5] = shadow_column[0] – shadow_column[5];

  column[6] = shadow_column[6] + shadow_column[1];
  column[1] = shadow_column[6] – shadow_column[1];

  return;
}

/*
 * IDCT():
 *      Perform 2D IDCT on a block
 * Input:
 *      REGISTER block[64]
 * Output:
 *      block
 * Return value:
 *      none
 */
void IDCT(REGISTER block[64])
{
  int i;

  Scale(block, -4);

  for (i=0; i<8; i++)
    Butterfly(block+8*i, 0);

  Transpose(block);

  for (i=0; i<8; i++)
    Butterfly(block+8*i, 1);

  Round(block, 6, -256, 255);

  Swap(block);
}
```

For informative purposes, a related forward discrete cosine transform (FDCT) implementation is shown below. This fixed-point FDCT does not form an integral part of this Recommendation.

```
/*****************************************************************************
 *
 *                            FIXED-POINT FDCT
 *
 * Fixed-point fast, separable fdct
 * Storage precision: 16 bits signed
 * Internal calculation precision: 32 bits signed
 * Input range: 9 bits signed, stored in 16 bits
 * Output range: [-2048, +2047]
 * All operations are signed
 *
 *****************************************************************************/
```

```
/*
 * Function declarations
 */

void FButterfly(REGISTER column[8]);
void FDCT(REGISTER block[64]);

/*
 * FButterfly():
 *      Perform 1D FDCT on a column
 * Input:
 *      REGISTER column[8]
 * Output:
 *      column
 * Return value:
 *      none
 */
void FButterfly(REGISTER column[8])
{
  int i;
  REGISTER shadow_column[8];

  /*
   * For readability, we use a shadow column
   * that contains the state of column at the
   * preceding stage of the butterfly.
   */

  /*
   * Initialization
   */

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * First Phase
   */

  for (i=0; i<4; i++) {
    column[i]   = shadow_column[i] + shadow_column[7-i];
    column[7-i] = shadow_column[i] – shadow_column[7-i];
  }

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * Second Phase
   */

  column[0] = shadow_column[0] + shadow_column[3];
  column[3] = shadow_column[0] – shadow_column[3];

  column[1] = shadow_column[1] + shadow_column[2];
  column[2] = shadow_column[1] – shadow_column[2];

  column[4] = Multiply(OoR2, shadow_column[4], -2);
  column[7] = Multiply(OoR2, shadow_column[7], -2);

  column[6] = shadow_column[6] – shadow_column[5];
  column[5] = shadow_column[6] + shadow_column[5];

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * Third Phase
   */
```

```
  column[0] = shadow_column[0] + shadow_column[1];
  column[1] = shadow_column[0] – shadow_column[1];

  column[6] = shadow_column[6] – shadow_column[4];
  column[4] = shadow_column[6] + shadow_column[4];

  column[7] = shadow_column[7] – shadow_column[5];
  column[5] = shadow_column[7] + shadow_column[5];

  for (i=0; i<8; i++)
    shadow_column[i] = column[i];

  /*
   * Fourth Phase
   */

  Rotate(column+2, column+3, -2, -1, cpo8, spo8, 0);
  Rotate(column+4, column+5, -1, -1, cpo16, spo16, 0);
  Rotate(column+6, column+7, -1, -1, c3po16, s3po16, 0);

  return;
}

/*
 * FDCT():
 *      Perform 2D FDCT on a block
 * Input:
 *      REGISTER block[64]
 * Output:
 *      block
 * Return value:
 *      none
 */
void FDCT(REGISTER block[64])
{
  int i;

  for (i=0; i<8; i++)
    FButterfly(block+8*i);

  Transpose(block);

  for (i=0; i<8; i++)
    FButterfly(block+8*i);

  Round(block, 3, -2048, 2047);

  Swap(block);
}
```

W.6 Picture message

The picture message function indicates the presence of one or more octets representing message data. The first octet of the message data is a message header with the following structure, as shown in Figure W.1.

Figure W.1/H.263 – Structure of first message octet

DSIZE shall be equal to the number of octets in the message data corresponding to a picture message function, including the first octet shown in Figure W.1.

Decoders shall parse picture message data as required by basic PSUPP syntax, but decoder response to picture messages is otherwise undefined.

W.6.1 Continuation (CONT) (1 bit)

If equal to "1", CONT indicates that the message data associated with this picture message function is part of the same logical message as the message data associated with the next picture message function. If equal to "0", CONT indicates that the message data associated with this picture message function terminates the current logical message. CONT may be used, for example, to represent logical messages that span more than 14 octets.

W.6.2 End Bit Position or Track Number (EBIT) (3 bits)

For non-text picture messages, EBIT specifies the number of least significant bits that shall be ignored in the last message octet. In non-text picture messages, if CONT is "1", or if there is only one message octet (i.e. the octet in Figure W.1), EBIT shall equal "0". The number of valid message bits for a non-text picture message function excluding the CONT/EBIT/MTYPE bits is equal to (DSIZE – 1) × 8 – EBITS. The number of valid message bits for a logical message may be greater due to continuation.

For picture message types containing text information, EBIT shall contain a text track number. The precise meaning of the text track number is not specified herein, but should indicate a particular type (e.g. language) for the text. Track number zero should be considered the default track.

W.6.3 Message Type (MTYPE) (4 bits)

MTYPE indicates the type of message. The defined types are shown in Table W.2.

Table W.2/H.263 – MTYPE message type values

0	Arbitrary Binary Data
1	Arbitrary Text
2	Copyright Text
3	Caption Text
4	Video Description Text
5	Uniform Resource Identifier Text
6	Current Picture Header Repetition
7	Previous Picture Header Repetition
8	Next Picture Header Repetition, Reliable TR
9	Next Picture Header Repetition, Unreliable TR
10	Top Interlaced Field Indication
11	Bottom Interlaced Field Indication
12	Picture Number
13	Spare Reference Pictures
14..15	Reserved

W.6.3.1 Arbitrary binary data

Arbitrary binary data is used to convey any non-ISO/IEC 10646-1 UTF-8 coded binary message. The interpretation of contents of the arbitrary binary data are outside the scope of this Recommendation, but should begin with some identifying pattern (e.g. a four-octet identifier code) to aid in distinguishing one type of such data from others.

W.6.3.2 Arbitrary text

Arbitrary text is used to convey a generic ISO/IEC 10646-1 UTF-8 coded text message. More specific text messages such as copyright information should be represented with other message types (e.g. copyright text) as appropriate.

W.6.3.3 Copyright text

Copyright text shall be used only to convey intellectual property information regarding the source or the encoded representation in the bitstream. The copyright message shall be coded according to ISO/IEC 10646-1 UTF-8.

W.6.3.4 Caption text

Caption text shall be used only to convey caption information associated with the current and subsequent pictures of the bitstream. The caption message shall be coded according to ISO/IEC 10646-1 UTF-8. The caption text shall be inserted in the bitstream as if it were to be displayed in a separate text area where new text is appended at the end of previous text and earlier text scrolled away from the point of insertion. The Form Feed (hexadecimal "0x000C") control code shall be used to indicate clearing of the visible text area. The End of Medium (hexadecimal "0x0019") control code shall be used to indicate "caption off" status. However, this Recommendation puts no restriction on how caption text is actually displayed and stored.

W.6.3.5 Video description text

Video description text shall be used only to convey descriptive information associated with the information contents of the current bitstream. The video description shall be coded according to ISO/IEC 10646-1 UTF-8. The video description text shall be inserted in the bitstream as if it were to be displayed in a separate text area where new text is appended at the end of previous text and earlier text scrolled away from the point of insertion. The Form Feed (hexadecimal "0x000C") control code shall be used to indicate clearing of the visible text area. The End of Medium (hexadecimal "0x0019") control code shall be used to indicate "description off" status. However, this Recommendation puts no restriction on how video description text is actually displayed and stored.

W.6.3.6 Uniform Resource Identifier (URI) text

The message consists of a uniform resource identifier (URI), as defined in IETF RFC 2396. The URI shall be coded according to ISO/IEC 10646-1 UTF-8.

W.6.3.7 Current picture header repetition

The picture header from the current picture is repeated in this message. The repeated bits exclude any supplemental enhancement information (PEI/PSUPP). All other bits up to the GOB or Slice layer should be included, subject to the limitations of W.4.

W.6.3.8 Previous picture header repetition

The picture header from the previously transmitted picture is repeated in this message. The repeated bits exclude the first two bytes of picture start code (PSC) and any supplemental enhancement information (PEI/PSUPP). All other bits up to the GOB or Slice layer should be included, subject to the limitations of W.4.

W.6.3.9 Next picture header repetition, reliable TR

The picture header from the next picture to be transmitted is repeated in this message. The repeated bits exclude the first two bytes of picture start code (PSC) and any supplemental enhancement information (PEI/PSUPP). All other bits up to the GOB or Slice layer should be included, subject to the limitations of W.4.

W.6.3.10 Next picture header repetition, unreliable TR

The picture header from the next picture to be transmitted is repeated in this message. The repeated bits exclude the first three bytes of picture header and any supplemental enhancement information (PEI/PSUPP). All other bits up to the GOB or Slice layer should be included, subject to the limitations of W.4. Any TR or ETR bits in the repeated picture header are not necessarily the same as the corresponding bits in the next picture header.

W.6.3.11 Interlaced field indications

In the case of interlaced field indications, the message consists of an indication of interlaced field coding. This indication does not affect the decoding process. However, it indicates that the current picture was not actually scanned as a progressive-scan picture. In other words, it indicates that the current coded picture contains only half of the lines of the full resolution source picture. DSIZE shall be 1, CONT shall be 0, and EBIT shall be 0 for interlaced field indications. In the case of interlaced field coding, each increment of the temporal reference denotes the time between the sampling of alternate half-picture fields of a picture, rather than the time between two complete pictures. In the case of a top interlaced field indication, the current picture contains the first (i.e. top), third, fifth, etc. lines of the complete picture. In the case of a bottom interlaced field indication, the current picture contains the second, fourth, sixth, etc. lines of the complete picture. When sending interlaced field indications, an encoder shall conform to the following conventions:

1) The encoder shall use a picture clock frequency (custom picture clock frequency, if necessary) such that each new field of the original source video corresponds to an increment of 1 in the temporal reference.

2) The encoder shall use a picture size (custom picture size, if necessary) such that the picture dimensions correspond to those of a single field.

3) The encoder shall use a pixel aspect ratio (custom pixel aspect ratio, if necessary) such that the full-height picture aspect ratio corresponds to the picture aspect ratio derived from the pixel aspect ratio of the single field represented by the current encoded picture.

Interlaced field scanning was introduced originally as an analog video compression technique. Although progressive picture scanning is generally regarded as superior for digital compression and display, the use of interlaced field scanning has persisted in many camera and display designs. Interlaced field coding (which can be implemented with lower delay than either interlaced full-picture coding or progressive-scan picture coding at half the interlaced field rate) is therefore supported by the indications herein.

An encoder shall not send interlaced field indications unless the capability of the decoder to receive and properly process such field-based pictures has been established by external means (for example, ITU-T H.245). Failure to establish such a decoder capability may produce a visually annoying small-amplitude vertical shaking behaviour in the decoded picture received and displayed by a decoder.

For example, an encoder may use interlaced field coding with application of the Reference Picture Selection mode (specified in Annex N) or the Enhanced Reference Picture Selection mode (specified in Annex U) to allow the addressing of more than one prior field. For "525/60" interlaced field coding for a 4:3 picture aspect ratio with 704 coded luminance samples per line and 240 coded luminance lines per field, the encoder shall use a custom picture size having a picture width of 704 and a picture height of 240, a custom pixel aspect ratio of 5:11, and a custom picture clock frequency specified with a clock conversion code "1" and a clock divisor of 30. For "625/50" interlaced field coding for a 4:3 picture aspect ratio with 704 coded luminance samples per line and 288 coded luminance lines per field, the encoder shall use a custom picture size having a picture width of 704 and a picture height of 288, a custom pixel aspect ratio of 6:11, and a custom picture clock frequency specified with a clock conversion code "0" and a clock divisor of 36.

The vertical sampling positions of the chrominance samples in interlaced field coding of a top field picture are specified as shifted up by 1/4 luminance sample height relative to the field sampling grid in order for these samples to align vertically to the usual position relative to the full-picture sampling grid. The vertical sampling positions of the chrominance samples in interlaced field coding of a bottom field picture are specified as shifted down by 1/4 luminance sample height relative to the field sampling grid in order for these samples to align vertically to the usual position relative to the full-picture sampling grid. The horizontal sampling positions of the chrominance samples are specified as unaffected by the application of interlaced field coding. The vertical sampling positions are shown with their corresponding temporal sampling positions in Figure W.2.

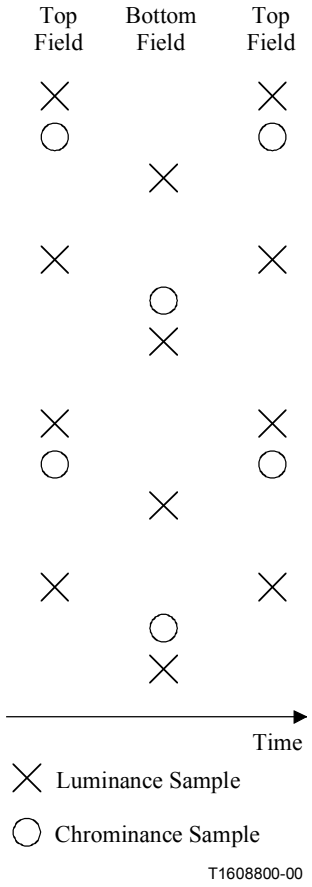

Figure W.2/H.263 – Vertical and temporal alignment of chrominance samples for interlaced field coding

W.6.3.12 Picture number

This message shall not be used if Annex U is in use. The message contains two data bytes that carry a 10-bit Picture Number. Consequently, DSIZE shall be 3, CONT shall be 0, and EBIT shall be 6. Picture Number shall be incremented by 1 for each coded and transmitted I or P picture or PB or Improved PB frame, in a 10-bit modulo operation. For EI and EP pictures, Picture Number shall be incremented for each EI or EP picture within the same scalability enhancement layer. For B pictures, Picture Number shall be incremented relative to the value in the most recent non-B picture in the reference layer of the B picture which precedes the B picture in bitstream order (a picture which is temporally subsequent to the B picture). If adjacent pictures in the same enhancement layer have the same temporal reference, and if the reference picture selection mode (see Annex N) is in use, the decoder shall regard this occurrence as an indication that redundant copies have been sent of approximately the same pictured scene content, and all of these pictures shall share the same Picture

Number. If the difference (modulo 1024) of the Picture Numbers of two consecutively received non-B pictures in the same enhancement layer is not 1, and if the pictures do not represent approximately the same pictured scene content as described above, the decoder should infer a loss of pictures or corruption of data.

W.6.3.13 Spare reference pictures

Encoders can use this message to instruct decoders which pictures resemble the current motion compensation reference picture so well that one of them can be used as a spare reference picture if the actual reference picture is lost during transmission. If a decoder lacks an actual reference picture but can access a spare reference picture, it should not request for an INTRA picture update. It is up to encoders to choose the spare reference pictures, if any. The message data bytes contain the Picture Numbers of the spare reference pictures in preference order (the most preferred appearing first). Picture Numbers refer to the values that are transmitted according to Annex U or W.6.3.12. This message can be used for P, B, PB, Improved PB, and EP picture types. However, if Annex N or Annex U is in use and if the picture is associated with multiple reference pictures, this message shall not be used. For EP pictures, the message shall be used only for forward prediction, whereas upward prediction is always done from the temporally corresponding reference layer picture. For B, PB, and Improved PB picture types, it specifies a picture for use as a forward motion prediction reference. This message shall not be used if the picture is an I or EI picture.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Geneva, 2001