UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

H.263
Annexe W
(11/2000)

SÉRIE H: SYSTÈMES AUDIOVISUELS ET MULTIMÉDIAS

Infrastructure des services audiovisuels – Codage des images vidéo animées

---

Codage vidéo pour communications à faible débit

Annexe W: Informations complémentaires d'amélioration additionnelles

Recommandation UIT-T H.263 – Annexe W

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE H

SYSTÈMES AUDIOVISUELS ET MULTIMÉDIAS

CARACTÉRISTIQUES DES SYSTÈMES VISIOPHONIQUES	H.100–H.199
INFRASTRUCTURE DES SERVICES AUDIOVISUELS	
Généralités	H.200–H.219
Multiplexage et synchronisation en transmission	H.220–H.229
Aspects système	H.230–H.239
Procédures de communication	H.240–H.259
Codage des images vidéo animées	**H.260–H.279**
Aspects liés aux systèmes	H.280–H.299
SYSTÈMES ET ÉQUIPEMENTS TERMINAUX POUR LES SERVICES AUDIOVISUELS	H.300–H.399
SERVICES COMPLÉMENTAIRES EN MULTIMÉDIA	H.450–H.499

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T H.263

Codage vidéo pour communications à faible débit

ANNEXE W

Informations complémentaires d'amélioration additionnelles

Résumé

D'autres *informations d'amélioration complémentaires*, qui peuvent facultativement être ajoutées à un flux binaire H.263 pour offrir des améliorations compatibles vers l'arrière, notamment:

- L'indication de l'utilisation d'une transformée IDCT à virgule fixe spécifique.
- Des messages d'image, notamment les types de message:
  - de données binaires arbitraires;
  - de texte (arbitraire, copyright, légende, description vidéo ou identificateur universel de ressource);
  - de répétition d'en-tête d'image (en cours, précédente, suivante avec référence temporelle fiable ou suivante avec référence temporelle non fiable);
  - d'indications de trames entrelacées (supérieure ou inférieure); et
  - d'identification d'image de référence de réserve.

Source

L'Annexe W de la Recommandation H.263 de l'UIT-T, élaborée par la Commission d'études 16 (2001-2004) de l'UIT-T, a été approuvée le 17 novembre 2000 selon la procédure définie dans la Résolution 1 de l'AMNT.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT avait été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Recommandation UIT-T H.263

Codage vidéo pour communications à faible débit

ANNEXE W

Informations complémentaires d'amélioration additionnelles

W.1 Domaine d'application

La présente annexe décrit le format d'autres informations d'amélioration complémentaires envoyées dans le champ PSUPP de la couche image H.263, qui s'ajoutent à la fonctionnalité définie à l'Annexe L. La capacité d'un décodeur de fournir tout ou partie des capacités décrites dans la présente annexe peut être signalée par des moyens séparés (ceux de l'UIT-T H.245 par exemple). Les décodeurs qui n'offrent pas les capacités additionnelles peuvent simplement éliminer les bits d'information PSUPP nouvellement définis qui apparaissent dans le flux binaire. La présence de ces informations d'amélioration complémentaires est indiquée par la présence du bit PEI et de l'octet PSUPP qui le suit dont le champ FTYPE a l'une des deux valeurs nouvellement définies. L'interprétation de base des valeurs de PEI, PSUPP, FTYPE et DSIZE est identique à celle donnée dans l'Annexe L et aux paragraphes 5.1.24 et 5.1.25.

W.2 Références

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée.

[8] ISO/CEI 10646-1:2000, *Technologies de l'information – Jeu universel de caractères codés sur plusieurs octets* (JUC) – *Partie 1: Architecture et plan multilingue de base*.

[9] IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax* (*Identificateurs universels de ressources: syntaxe générique*).

W.3 Valeurs FTYPE additionnelles

Deux valeurs qui étaient réservées dans le Tableau L.1 de l'Annexe L sont définies dans le Tableau W.1.

Tableau W.1/H.263 – Valeurs de type de fonction FTYPE

13	Transformation IDCT à virgule fixe
14	Message d'image

W.4 Nombre maximal recommandé d'octets PSUPP

Lorsqu'on utilise l'une des fonctions FTYPE précédemment mentionnées et définies dans la présente annexe, le nombre total d'octets PSUPP par image doit être maintenu à une valeur raisonnablement faible par rapport à la taille de l'image codée et ne doit pas dépasser 256 octets, quelle que soit la taille de l'image codée.

NOTE – Certains protocoles de transmission de données utilisés pour l'acheminement du flux binaire vidéo peuvent prévoir la répétition externe du contenu de l'en-tête d'image à des fins de résistance aux erreurs et imposer des limites à la quantité de données de ce type pouvant être répétées depuis un en-tête d'image (par exemple 504 bits dans le format de mise en paquets IETF RFC 2429). L'introduction d'un grand nombre d'octets PSUPP peut se traduire par incapacité d'un tel protocole externe de permettre la répétition complète du contenu de l'en-tête de l'image.

W.5 Transformation IDCT à virgule fixe

La fonction IDCT à virgule fixe indique qu'une approximation IDCT particulière est utilisée dans la construction du flux binaire. DSIZE doit être égal à 1 pour la fonction IDCT à virgule fixe. L'octet de données PSUPP qui suit spécifie la réalisation particulière de la fonction IDCT. La valeur zéro indique l'IDCT 0 de référence décrite en W.5.3; les valeurs 1 à 255 sont réservées.

W.5.1 Fonctionnement du décodeur

La capacité d'un décodeur d'effectuer une transformation IDCT donnée à virgule fixe peut être signalée au codeur par des moyens séparés (ceux de l'UIT-T H.245, par exemple). Lorsqu'il reçoit un flux binaire codé avec l'indication IDCT à virgule fixe, un décodeur doit utiliser l'IDCT à virgule fixe en question s'il a la capacité de le faire.

W.5.2 Suppression du rafraîchissement forcé

L'Annexe A spécifie les prescriptions de précision pour la transformation en cosinus discrète inverse (IDCT) tout en permettant de nombreuses mises en œuvre conformes. Pour limiter l'accumulation d'erreurs dues à des transformations IDCT discordantes au niveau du codeur et au niveau du décodeur, le paragraphe 4.4 ("Rafraîchissement forcé") nécessite de coder les macroblocs en mode INTRA au moins une fois toutes les 132 fois où les coefficients sont transmis.

Si le type de fonction IDCT à virgule fixe est indiqué dans le flux binaire, la nécessité du rafraîchissement forcé est supprimée et la fréquence des codages INTRA n'est pas réglementée. Le codeur doit néanmoins poursuivre le rafraîchissement forcé à moins qu'il n'ait établi par des moyens séparés que le décodeur a la capacité nécessaire pour la transformation IDCT à virgule fixe spécifiée; sinon, il peut y avoir discordance.

W.5.3 IDCT 0 de référence

L'IDCT 0 de référence correspond à toute implémentation qui, pour chaque bloc d'entrée, produit les mêmes valeurs de sortie que le programme source C reproduit ci-dessous.

NOTE – L'IDCT à virgule fixe est conforme à l'Annexe A/H.263, mais pas à la prescription relative à l'intervalle étendu de valeurs de l'Annexe A de la Rec. UIT-T H.262 | ISO/CEI 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);
}
```

Une implémentation correspondante de transformation en cosinus discrète directe (FDCT, *forward discrete cosine transform*) est montrée ci-dessous à titre d'information. Cette FDCT à virgule fixe ne fait pas partie intégrante de la présente Recommandation.

```
/*****************************************************************************
 *
 *                            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 Message d'image

La fonction de message d'image indique la présence d'un ou de plusieurs octets représentant des données de message. Le premier octet de ces données de message est un en-tête de message qui présente la structure suivante, comme décrit dans la Figure W.1

Figure W.1/H.263 – Structure du premier octet du message

DSIZE doit être égal au nombre d'octets de données de message qui correspondent à une fonction de message d'image, y compris le premier octet montré à la Figure W.1.

Les décodeurs doivent analyser les données du message d'image conformément aux prescriptions de la syntaxe PSUPP de base, mais hormis cela la réponse du décodeur aux messages d'image n'est pas définie.

W.6.1 Continuation (CONT) (1 bit)

S'il a la valeur "1", CONT indique que les données de message associées à cette fonction de message d'image font partie du même message logique que les données de message associées à la fonction de message de l'image suivante. S'il a la valeur "0", CONT indique que les données de message associées à cette fonction de message d'image terminent le message logique en cours. CONT peut être utilisé, par exemple, pour représenter les messages logiques qui occupent plus de 14 octets.

W.6.2 Position du bit final ou numéro de désignateur (EBIT) (3 bits)

Pour les messages d'image sans texte, EBIT spécifie le nombre de bits de plus faible poids qu'il faut ignorer dans le dernier octet du message. Si, dans des messages d'image sans texte, CONT a la valeur "1", ou s'il n'y a qu'un octet de message (c'est-à-dire l'octet de la Figure W.1), EBIT sera égal à "0". Le nombre de bits de message valables pour une fonction de message d'image sans texte, non compris les bits CONT/EBIT/MTYPE, est égal à (DSIZE − 1) × 8 − EBITS. Le nombre de bits de message valables pour un message logique peut être supérieur en raison de la continuation.

En ce qui concerne les types de message d'image contenant des informations de texte, EBIT doit contenir un numéro de désignateur de texte. Le sens exact du numéro de désignateur de texte n'est pas spécifié ici, mais devrait indiquer un type particulier de texte (par exemple la langue). Il convient de considérer que le désignateur ayant le numéro 0 est le désignateur par défaut.

W.6.3 Type de message (MTYPE) (4 bits)

MTYPE désigne le type de message. Les types définis sont montrés dans le Tableau W.2.

Tableau W.2/H.263 – Valeurs du type de message MTYPE

0	Données binaires arbitraires
1	Texte arbitraire
2	Texte de copyright
3	Texte de légende
4	Texte de description vidéo
5	Texte d'identificateur universel de ressource
6	Répétition d'en-tête d'image en cours
7	Répétition d'en-tête d'image précédente
8	Répétition d'en-tête d'image suivante, référence TR fiable
9	Répétition d'en-tête d'image suivante, référence TR non fiable
10	Indication de trame entrelacée supérieure
11	Indication de trame entrelacée inférieure
12	Numéro d'image
13	Images de référence de réserve
14..15	Réservé

W.6.3.1 Données binaires arbitraires

Ces données sont utilisées pour acheminer tout message binaire qui n'est pas codé ISO/CEI 10646-1 UTF-8. L'interprétation du contenu des données binaires arbitraires ne relève pas du domaine de la présente Recommandation, mais ces données devraient commencer par un élément d'identification (par exemple un code identificateur à 4 octets) pour aider à faire la distinction entre ce type de données et d'autres.

W.6.3.2 Texte arbitraire

Le texte arbitraire est utilisé pour acheminer un message de texte générique codé ISO/CEI 10646-1 UTF-8. Les messages de texte plus spécifiques – informations de copyright par exemple – devraient être représentés au moyen d'autres types de message (texte de copyright, par exemple), selon qu'il convient.

W.6.3.3 Texte de copyright

Le texte de copyright doit être utilisé uniquement pour acheminer des informations de propriété intellectuelle relatives à la source ou à la représentation codée dans le flux binaire. Le message de copyright doit être codé conformément à l'ISO/CEI 10646-1 UTF-8.

W.6.3.4 Texte de légende

Le texte de légende doit être utilisé uniquement pour acheminer des informations de légende associées à l'image en cours et aux images subséquentes dans le flux binaire. Le message de légende doit être codé ISO/CEI 10646-1 UTF-8. Le texte de légende doit être introduit dans le flux binaire comme s'il devait être affiché dans une zone de texte distincte dans laquelle le nouveau texte est ajouté à la fin du texte précédent et on fait défiler, à partir du point d'insertion, le texte antérieur pour le faire disparaître. Le code de commande de saut de page ("0x000C" hexadécimal) doit être utilisé pour indiquer l'effacement de la zone de texte visible. Le code de commande de fin de média ("0x0019" hexadécimal) doit être utilisé pour indiquer l'état "fin de légende". Toutefois, la présente Recommandation n'impose aucune restriction quant à la manière dont le texte de légende est effectivement affiché et enregistré.

W.6.3.5 Texte de description vidéo

Le texte de description vidéo doit être utilisé uniquement pour acheminer des informations descriptives associées au contenu informatif du flux binaire en cours. La description vidéo doit être codée ISO/CEI 10646-1 UTF-8. Le texte de description vidéo doit être introduit dans le flux binaire comme s'il devait être affiché dans une zone de texte distincte dans laquelle le nouveau texte est ajouté à la fin du texte précédent et on fait défiler, à partir du point d'insertion, le texte antérieur pour le faire disparaître. Le code de commande de saut de page ("0x000C" hexadécimal) doit être utilisé pour indiquer l'effacement de la zone de texte visible. Le code de commande de fin de média ("0x0019" hexadécimal) doit être utilisé pour indiquer l'état "fin de description". Toutefois, la présente Recommandation n'impose aucune restriction quant à la manière dont le texte de description vidéo est effectivement affiché et enregistré.

W.6.3.6 Texte d'identificateur universel de ressource (URI, *uniform resource identifier*)

Le message consiste en un identificateur universel de ressource (URI), tel que défini dans l'IETF RFC 2396. L'URI doit être codé ISO/CEI 10646-1 UTF-8.

W.6.3.7 Répétition d'en-tête d'image en cours

L'en-tête de l'image en cours est répété dans ce message. Les bits répétés excluent toute information d'amélioration complémentaire (PEI/PSUPP). Tous les autres bits jusqu'à la couche GOB ou la couche tranche doivent être inclus, sous réserve des limitations énoncées en W.4.

W.6.3.8 Répétition d'en-tête d'image précédente

L'en-tête de l'image transmise précédemment est répété dans ce message. Les bits répétés excluent les deux premiers octets du code de début de l'image (PSC, *picture start code*) et toute information d'amélioration complémentaire (PEI/PSUPP). Tous les autres bits jusqu'à la couche GOB ou la couche tranche doivent être inclus, sous réserve des limitations énoncées en W.4.

W.6.3.9 Répétition d'en-tête d'image suivante, référence TR fiable

L'en-tête de l'image suivante à transmettre est répété dans ce message. Les bits répétés excluent les deux premiers octets du code de début de l'image (PSC) et toute information d'amélioration complémentaire (PEI/PSUPP). Tous les autres bits jusqu'à la couche GOB ou la couche tranche doivent être inclus, sous réserve des limitations énoncées en W.4.

W.6.3.10 Répétition d'en-tête d'image suivante, référence TR non fiable

L'en-tête de l'image suivante à transmettre est répété dans ce message. Les bits répétés excluent les trois premiers octets de l'en-tête d'image et toute information d'amélioration complémentaire (PEI/PSUPP). Tous les autres bits jusqu'à la couche GOB ou la couche tranche doivent être inclus, sous réserve des limitations énoncées en W.4. Tous les bits TR ou ETR dans l'en-tête d'image répété ne sont pas nécessairement les mêmes que les bits correspondants de l'en-tête d'image suivante.

W.6.3.11 Indications de trames entrelacées

Dans le cas des indications de trames entrelacées, le message est constitué d'une indication de codage à trames entrelacées. Celle-ci n'affecte pas le processus de décodage. Toutefois, elle indique que l'image en cours n'a pas été balayée au moyen d'un balayage progressif; autrement dit, elle indique que l'image codée en cours ne contient que la moitié des lignes de l'image source complète. Pour les indications de trames entrelacées, DSIZE doit être à 1, CONT à 0 et EBIT à 0. Dans le cas du codage à trames entrelacées, chaque incrément de la référence temporelle correspond au temps entre l'échantillonnage des trames de demi-images alternées et non le temps entre deux images complètes. Dans le cas d'une indication de trame entrelacée supérieure, l'image en cours contient la première (c'est-à-dire supérieure), troisième, cinquième, etc. lignes de l'image complète. Dans le cas d'une indication de trame entrelacée inférieure, l'image en cours contient la deuxième, quatrième, sixième, etc. lignes de l'image complète. Lorsqu'il envoie des indications de trames entrelacées, un codeur doit se conformer aux conventions suivantes:

1) utiliser une fréquence d'horloge d'image (personnalisée au besoin) telle que chaque nouvelle trame de la source vidéo d'origine corresponde à un incrément de la référence temporelle;
2) utiliser une taille d'image (personnalisée au besoin) telle que les dimensions d'image correspondent à celles d'une seule trame;
3) utiliser un format en pixels (personnalisé au besoin) tel que le format de l'image complète corresponde au format d'image obtenu à partir du format en pixels de la trame unique représentée par l'image codée en cours.

Le balayage à trames entrelacées a été introduit au départ comme une technique de compression vidéo analogique. Bien que le balayage progressif soit généralement jugé supérieur pour la compression numérique et l'affichage, le balayage à trames entrelacées est toujours couramment utilisé à la prise de vue et à l'affichage. Pour cette raison, le codage à trames entrelacées (qui peut être appliqué plus rapidement que le codage de l'image complète avec entrelacement ou le codage d'image par balayage progressif à la moitié de la fréquence de trames entrelacées) est pris en charge selon les indications données dans la présente annexe.

Un codeur ne doit pas envoyer d'indication de trames entrelacées à moins qu'il n'ait été établi, par des moyens séparés (ceux de l'UIT-T H.245, par exemple) que le décodeur a la capacité de recevoir et de traiter correctement de telles images basées sur des trames. A défaut, l'image décodée qui est reçue et affichée par le décodeur peut être sujette à des tremblements verticaux de faible amplitude entraînant une gêne visuelle.

Par exemple, un codeur peut utiliser le codage à trames entrelacées en appliquant le mode de sélection d'image de référence (spécifié à l'Annexe N) ou le mode de sélection d'image de référence amélioré (spécifié à l'Annexe U) afin de permettre l'adressage de plusieurs trames précédentes. Pour le codage à trames entrelacées "525/60" correspondant à un format 4:3 avec 704 échantillons de luminance codés par ligne et 240 lignes de luminance codées par trame, le codeur doit utiliser une taille d'image personnalisée ayant une largeur de 704 et une hauteur de 240, un format personnalisé de 5:11 et une fréquence d'horloge d'image personnalisée spécifiée au moyen d'un code de conversion d'horloge "1" et d'un diviseur d'horloge de 30. Pour le codage à trames entrelacées "625/50" correspondant à un format 4:3 avec 704 échantillons de luminance codés par ligne et 288 lignes de luminance codées par trame, le codeur doit utiliser une taille d'image personnalisée ayant une largeur de 704 et une hauteur de 288, un format personnalisé de 6:11 et une fréquence d'horloge d'image personnalisée spécifiée au moyen d'un code de conversion d'horloge "0" et d'un diviseur d'horloge de 36.

Les positions d'échantillonnage verticales des échantillons de chrominance dans le codage à trames entrelacées d'une image de trame supérieure sont spécifiées comme étant décalées vers le haut d'1/4 de hauteur d'échantillon de luminance par rapport à la grille d'échantillonnage de trame afin que ces échantillons s'alignent verticalement sur la position habituelle par rapport à la grille d'échantillonnage de l'image complète. Les positions d'échantillonnage verticales des échantillons de chrominance dans le codage à trames entrelacées d'une image de trame inférieure sont spécifiées comme étant décalées vers le bas d'1/4 de hauteur d'échantillon de luminance par rapport à la grille d'échantillonnage de trame afin que ces échantillons s'alignent verticalement sur la position habituelle par rapport à la grille d'échantillonnage de l'image complète. Il est spécifié que les positions d'échantillonnage horizontales des échantillons de chrominance ne sont pas touchées par l'application du codage à trames entrelacées. Les positions d'échantillonnage verticales sont montrées avec leurs positions d'échantillonnage temporelles correspondantes à la Figure W.2.

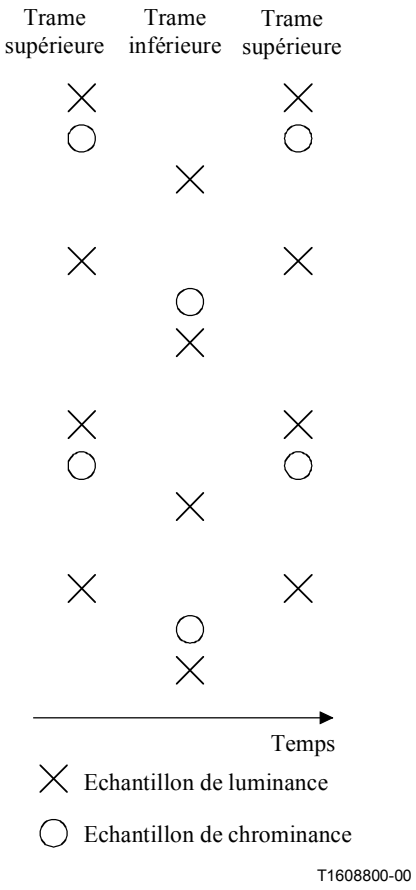

Figure W.2/H.263 – Alignement vertical et temporel des échantillons de chrominance pour le codage à trames entrelacées

W.6.3.12 Numéro d'image

Ce message ne sera pas utilisé en cas d'utilisation de la méthode de l'Annexe U. Il contient deux octets de données qui acheminent un numéro d'image à 10 bits. En conséquence, DSIZE aura la valeur 3, CONT la valeur 0 et EBIT la valeur 6. Le numéro d'image doit être incrémenté de 1 pour chaque image I, P, PB ou PB améliorée codée et transmise, dans une opération modulo à 10 bits. Pour les images EI et EP, le numéro d'image doit être incrémenté pour chaque image EI ou EP dans la même couche d'amélioration d'échelonnabilité. Dans le cas des images B, le numéro d'image doit être incrémenté par rapport à la valeur de l'image non-B la plus récente dans la couche de référence de l'image B qui précède l'image B dans l'ordre du flux binaire (une image qui est temporairement subséquente à l'image B). Si des images adjacentes dans la même couche d'amélioration ont la même référence temporelle, et si le mode de sélection d'image de référence (Annexe N) est utilisé, le décodeur doit considérer cette occurrence comme une indication que des copies redondantes contenant approximativement la même image ont été envoyées, et toutes ces images doivent partager le même numéro d'image. Si la différence (modulo 1024) entre les numéros de deux images non-B reçues consécutivement dans la même couche d'amélioration est différente de 1 et si les images ne représentent pas approximativement le même contenu que celui décrit ci-dessus, le décodeur doit conclure à une perte d'image ou une dégradation des données.

W.6.3.13 Images de référence de réserve

Le codeur peut utiliser ce message pour indiquer au décodeur quelles sont les images dont la ressemblance avec l'image de référence avec compensation de mouvement est telle que l'une d'elles peut être utilisée comme image de référence de réserve si l'image de référence effective se perd pendant la transmission. Si une image de référence effective manque au niveau du décodeur et que

celui-ci peut accéder à une image de référence de réserve, il ne devrait pas demander de rafraîchissement d'image INTRA. C'est le codeur qui doit choisir les éventuelles images de référence de réserve. Les octets de données du message contiennent les numéros des images de référence de réserve par ordre de préférence (l'image ayant la préférence la plus grande apparaissant en premier). Les numéros d'image renvoient aux valeurs qui sont transmises conformément à l'Annexe U ou au W.6.3.12. Ce message peut être utilisé pour les types d'image P, B, PB, PB amélioré et EP. Toutefois, si l'Annexe N ou l'Annexe U est utilisée et si l'image est associée à plusieurs images de référence, il ne faut pas utiliser ce message. Dans le cas des images EP, le message doit être utilisé uniquement pour la prédiction directe, alors que la prédiction inverse est toujours effectuée à partir de l'image de la couche de référence temporellement correspondante. Dans le cas des images de type B, PB et PB amélioré, le message spécifie l'image destinée à être utilisée comme référence pour la prédiction directe avec compensation de mouvement. Ce message ne doit pas être utilisé dans le cas d'une image I ou EI.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	**Systèmes audiovisuels et multimédias**
Série I	Réseau numérique à intégration de services
Série J	Transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	Langages et aspects informatiques généraux des systèmes de télécommunication

Genève, 2001