RECOMMANDATION UIT-R BR.1352

FORMAT DES FICHIERS POUR L'ÉCHANGE DE PROGRAMMES AUDIO SUR SUPPORTS INFORMATIQUES

(Question UIT-R 215/10)

(1998)

L'Assemblée des radiocommunications de l'UIT,

considérant

a) que les supports d'enregistrement fondés sur l'informatique, y compris les disques et les bandes de données, sont appelés à pénétrer dans tous les domaines de la production audio pour la radiodiffusion, à savoir l'édition non linéaire, la restitution à l'antenne et l'archivage;

b) que l'informatique offre de notables avantages en termes de souplesse d'exploitation, de flux de production et d'automatisation des stations; qu'elle est donc intéressante pour la modernisation de studios existants et pour la conception de nouvelles installations de studio;

c) que l'adoption d'un unique format de fichier pour l'échange de signaux simplifierait beaucoup l'interfonctionnement d'équipements individuels et de studios distants tout en facilitant l'intégration souhaitable des opérations d'édition, de restitution à l'antenne et d'archivage;

d) qu'un ensemble minimal d'informations relatives à la diffusion doit être inclus dans le fichier afin de décrire le signal audio;

e) que, pour assurer la compatibilité entre applications de complexités diverses, il faut adopter un ensemble minimal de fonctions communes à toutes les applications capables de traiter le format de fichier recommandé;

f) que la Recommandation UIT-R BS.646 définit le format audionumérique utilisé en production audio pour la diffusion radiophonique et télévisuelle;

g) que divers formats multicanaux font l'objet de la Recommandation UIT-R BS.775 et que ces formats sont appelés à être largement utilisés dans le proche avenir;

h) que la nécessité d'échanger des données audio apparaîtra également lorsque les systèmes de codage selon l'ISO/CEI 11172-3 et 13818-3 seront utilisés pour comprimer le signal;

j) que plusieurs diffuseurs mondiaux se sont déjà entendus afin d'adopter un format commun pour l'échange de fichiers de programme;

k) que la comptabilité avec les formats de fichiers actuellement disponibles sur le marché pourrait minimiser les efforts de l'industrie en vue de mettre en oeuvre le présent format dans les équipements,

recommande

1 que, pour l'échange de programmes audio sur supports informatiques, les paramètres, la fréquence d'échantillonnage, la résolution de codage et la préaccentuation du signal audio soient déterminés conformément aux parties applicables de la Recommandation UIT-R BS.646;

2 que le format de fichier spécifié dans l'Annexe 1 soit utilisé pour l'échange[1] de programmes audio en format MIC linéaire sur supports informatiques;

3 que, lorsque les signaux audio sont codés selon les systèmes ISO/CEI 11172-3 ou 13818-3, le format de fichier spécifié dans l'Annexe 1, complétée par l'Annexe 2, soit utilisé pour l'échange de programmes audio sur supports informatiques[2].

---

[1] L'adoption du format de fichier recommandé, non seulement pour les transferts mais aussi pour l'enregistrement sur le support, serait une solution préférable pour les utilisateurs car aucune servitude ne serait requise, aussi bien en terme de temps qu'en terme d'espace temporaire, pour la conversion de ces signaux audio lors de leur transfert entre les équipements. Il est toutefois reconnu qu'une recommandation dans ce sens pourrait pénaliser les développeurs qui utilisent certaines plates-formes informatiques.

[2] D'autres annexes pourront être définies ultérieurement afin d'élargir le format de fichier au transport de signaux audio codés par d'autres systèmes intéressant les diffuseurs.

ANNEXE 1

Spécification du format d'onde de radiodiffusion - Format de diffusion des fichiers de données audio

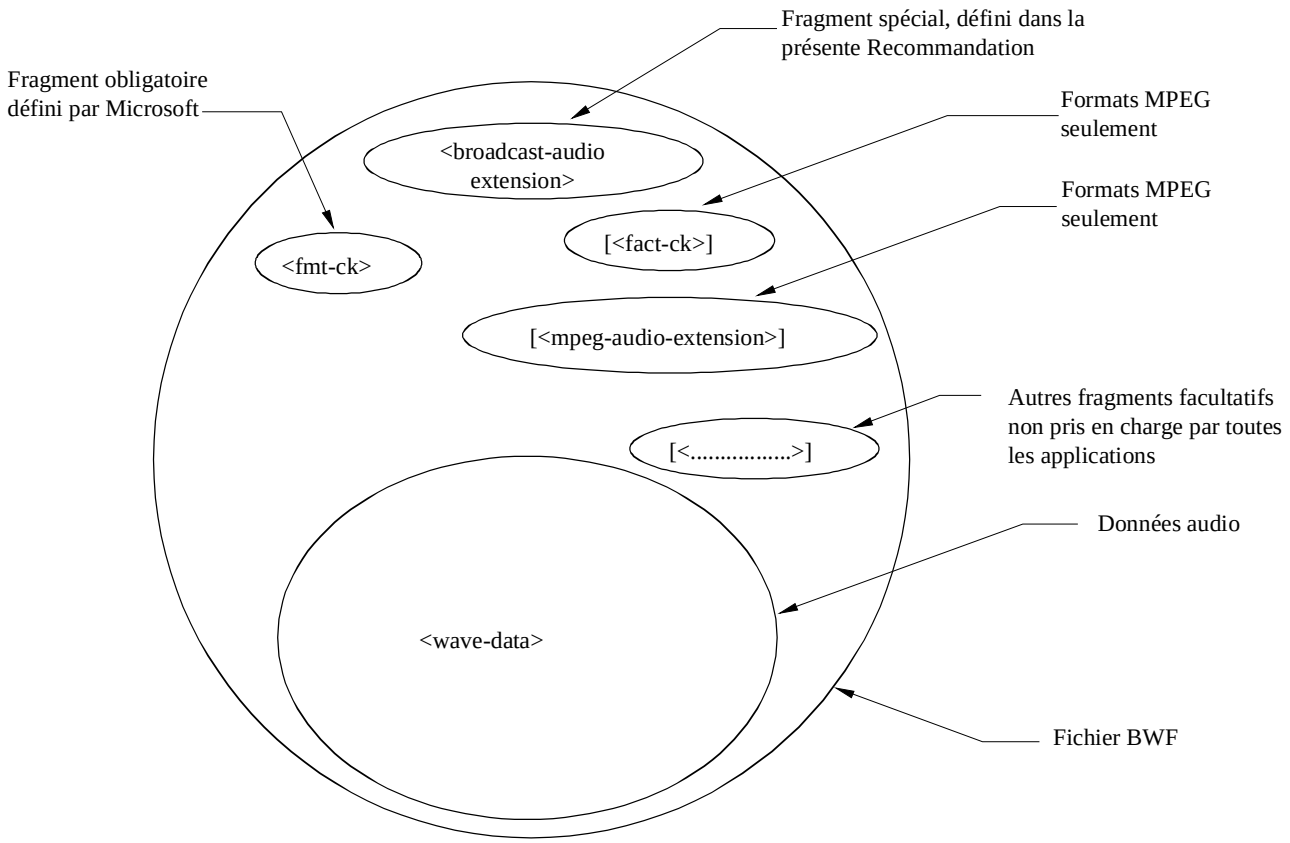
1 Introduction

Le format d'onde de radiodiffusion (BWF, *broadcast wave format*) est fondé sur le fichier audio WAVE de *Microsoft*®, qui est d'un type spécifié dans le format RIFF (*Resource Interchange File Format*) de cette compagnie. Les fichiers WAVE contiennent spécifiquement des données audio. Le module de construction fondamental du format de fichier RIFF, appelé *fragment*, contient un groupe d'éléments informationnels étroitement associés. Il est composé d'un identificateur de fragment, d'une valeur d'entier représentant la longueur en octets et des informations audio. Un fichier au format RIFF se compose d'un ensemble de fragments.

Pour le format BWF, certaines restrictions sont appliquées au format WAVE original. De plus, le fichier BWF comporte un fragment particulier: <broadcast-audio-extension>, comme illustré sur la Figure 1 ci-dessous.

FIGURE 1

Fichier au format BWF



Temp 10-11/33-01

La présente annexe spécifie le fragment d'extension audio pour la radiodiffusion qui est utilisé dans tous les fichiers BWF. Par ailleurs, l'appendice donne des renseignements sur le format RIFF de base et sur la façon dont il peut être étendu à d'autres types de données audio. Les détails du format audio à codage MIC sont également donnés dans l'appendice. Les spécifications particulières de l'extension à d'autres types de données audio seront publiées dans d'autres annexes de la présente Recommandation.

2 Fichiers au format BWF

2.1 Contenu d'un fichier BWF

Un fichier au format BWF doit commencer par l'en-tête "WAVE" du format RIFF de *Microsoft*® et doit comporter au moins les fragments suivants:

<WAVE-form> ->

RIFF('WAVE'

<broadcast_audio_extension>	//Information sur la séquence audio
<fmt-ck>	//Format du signal audio: MIC/MPEG
[<fact-ck>]	//Le fragment "fact" n'est requis que pour les formats MPEG
[<mpeg_audio_extension>]	//Le fragment "mpeg_audio_extension" n'est requis que pour les formats MPEG
<wave-data>)	//Données audio

NOTE – Tous les types additionnels de fragments qui sont présents dans le fichier doivent être considérés comme privés. Les applications ne sont pas tenues d'interpréter ou d'utiliser ces fragments. L'intégrité des données contenues dans des fragments non énumérés ci-dessus n'est donc pas garantie. Il y a cependant lieu que les applications au format BWF transmettent ces fragments chaque fois que possible.

2.2 Fragments déjà définis comme éléments de la norme RIFF

La norme RIFF est définie dans des documents publiés par *Microsoft Corporation*. La présente application utilise un certain nombre de fragments qui y sont déjà définis. Il s'agit des suivants:

fmt-ck

fact-ck

Les descriptions actuelles de ces fragments sont données pour information dans l'Appendice 1 de l'Annexe 1.

2.3 Fragment d'extension audio pour la radiodiffusion

Les paramètres supplémentaires qui sont nécessaires pour l'échange de programmes entre radiodiffuseurs sont ajoutés dans un fragment spécifique "d'extension audio pour la radiodiffusion", défini comme suit:

```
broadcast_audio_extension typedef struct {
    DWORD   ckID;           /* (broadcastextension)ckID=bext.      */
    DWORD   ckSize;         /* fragment de taille d'extension      */
    BYTE    ckData[ckSize]; /* données du fragment                 */
}
typedef struct broadcast_audio_extension {
CHAR Description[256];          /* ASCII : «Description de la séquence sonore»*/
CHAR Originator[32];            /* ASCII : «Nom de la source»*/
CHAR OriginatorReference[32];   /* ASCII : «Référence de la source»*/
CHAR OriginationDate[10];       /* ASCII : «yyyy:mm:dd» (année/mois/jour)*/
CHAR OriginationTime[8];        /* ASCII : «hh:mm:ss» (heure, minute, seconde)*/
DWORD TimeReferenceLow;         /*Premier comptage d'échantillons depuis minuit, mot faible*/
DWORD TimeReferenceHigh;        /* Premier comptage d'échantillons depuis minuit, mot fort*/
WORD Version;                   /* Version du format BWF; nombre binaire non signé */
CHAR Reserved[254] ;            /* Champ réservé pour usage futur, mis à "NULL" */
CHAR CodingHistory[];           /* ASCII : «Chronologie des codages» */
} BROADCAST_EXT
```

Champ	Description
Description	Chaîne de caractères ASCII (256 max.) donnant une description libre de la séquence. Pour faciliter les applications qui n'affichent qu'une courte description, il est recommandé d'insérer un résumé de la description dans les 64 premiers caractères, les 192 caractères suivants étant utilisés pour les détails. Si la longueur de la chaîne est inférieure à 256 caractères, le dernier d'entre eux est suivi d'un caractère “néant” (00).
Originator	Chaîne de caractères ASCII (32 max.) contenant le nom de la source/du producteur du fichier audio. Si la longueur de cette chaîne est inférieure à 32 caractères, le champ est terminé par un caractère “néant”.
OriginatorReference	Chaîne de caractères ASCII (32 max.) donnant une référence univoque, attribuée par l'organisation source. Si la longueur de cette chaîne est inférieure à 32 caractères, le champ est terminé par un caractère "néant".
OriginationDate10	Chaîne de caractères ASCII indiquant la date de la création de la séquence audio. Le format est «‘,année,’- ‘,mois, ’ - ‘,jour, ’» avec 4 caractères pour l'année et 2 caractères par autre élément. L'année est définie par un nombre compris entre 0000 et 9999 Le mois est défini par un nombre compris entre 1 et 12 Le jour est défini par un nombre compris entre 1 et 31 Le séparateur placé entre les éléments peut être quelconque mais il est recommandé d'utiliser l'un des caractères suivants: ‘-‘ tiret ‘_’ soulignement ‘:’ deux points ‘ ’ espace ‘.’ point
OriginationTime	Chaîne de 8 caractères ASCII indiquant l'heure de création de la séquence audio. Le format est «’,heure,’ - ‘,minute,’ - ‘,seconde,’» avec 2 caractères par élément. L'heure est définie par un nombre compris entre 0 et 23. La minute et la seconde sont définies par un nombre compris entre 0 et 59. Le séparateur placé entre les éléments peut être quelconque mais il est recommandé d'utiliser l'un des caractères suivants: ‘-‘ tiret ‘_’ soulignement ‘:’ deux points ‘ ’ espace ‘.’ point
TimeReference	Ce champ contient le code temporel de la séquence. C'est une valeur codée sur 64 bits qui contient le premier comptage d'échantillons depuis minuit. Le nombre d'échantillons par seconde dépend de la fréquence d'échantillonnage, qui est définie dans le champ <nSamplesPerSec> issu du fragment de format <**format chunk**>.
Version	Nombre binaire non signé qui indique la version du fichier BWF. Ce nombre est initialement mis à zéro.
Reserved	Chaîne de 254 octets réservés pour une extension. Ces 254 octets doivent être mis à une valeur NULL. Dans l'avenir, la valeur NULL sera utilisée comme défaut afin d'assurer la compatibilité.
CodingHistory	Caractères ASCII non restreints formant un ensemble de chaînes terminées par la séquence CR/LF. Chaque chaîne contient une description du processus de codage appliqué. Chaque nouvelle application de codage est appelée à ajouter une nouvelle chaîne contenant l'information appropriée. Cette information doit indiquer le type de son (MIC ou MPEG), avec ses paramètres spécifiques. MIC: mode (mono, stéréo), longueur d'échantillon (8 ou 16 bits) et fréquence d'échantillonnage. MPEG: fréquence d'échantillonnage, débit binaire, couche (I ou II) et mode (mono, stéréo, stéréo mixte ou son bicanal). Il est recommandé que les constructeurs de codeurs fournissent une chaîne ASCII à insérer dans le champ *CodingHistory*.

NOTE – Des études sont en cours pour proposer un format pour la chronologie de codage qui simplifieront l'interprétation de l'information fournie dans ce champ.

2.4 Autres informations propres aux applications

Des études sont en cours pour définir d'autres fragments transportant ou désignant des données propres à certaines applications, par exemple pour l'édition audio ou l'archivage.

APPENDICE (DE L'ANNEXE 1)

Format RIFF des fichiers WAVE (.WAV)

Les informations contenues dans le présent appendice sont extraites des spécifications du format de fichier RIFF de *Microsoft*®. Elles ne figurent ici qu'à titre documentaire.

Pour de plus amples informations, voir la plus récente version de l'utilitaire *Software Developers Kit* de *Microsoft*®, mis à jour pour les normes multimédias (Rév. 3.0 du 15 avril 1994 ou plus récent).

1 Format des fichiers audio de type WAVE

La forme WAVE est définie comme suit. Les programmes doivent attendre (et ignorer) tous les fragments inconnus qui seront rencontrés, comme avec toutes les formes de type RIFF. Le fragment <fmt-ck> doit cependant apparaître toujours avant le fragment <wave-data> et ces deux fragments sont obligatoires dans un fichier WAVE.

<WAVE-form> ->

```
RIFF ( 'WAVE'
        <fmt-ck>                    // Fragment de format
        [<fact-ck>]                 // Fragment factuel
        [<other-ck>]                // Autres fragments facultatifs
        <wave-data> )               // Données audio
```

Les fragments WAVE sont décrits dans les paragraphes suivants:

1.1 Fragment de format WAVE

Le fragment de format WAVE <fmt-ck> spécifie le format des données audio <wave-data>. Il est défini comme suit:

<fmt-ck> -> fmt(<common-fields>

<format-specific-fields>)

<common-fields> ->

```
struct{
        WORD wFormatTag;            //Catégorie de format
        WORD nChannels;             // Nombre de voies
        DWORD nSamplesPerSec;       // Fréquence d'échantillonnage
        DWORD nAvgBytesPerSec;      // Estimation du tampon
        WORD nBlockAlign;           // Longueur de bloc de données
}
```

Les champs contenus dans la portion <common-fields> du fragment sont les suivants:

Champ	Description
wFormatTag	Nombre indiquant la catégorie de format WAVE du fichier. Le contenu de la portion <format-specific-fields> du fragment 'fmt' ainsi que l'interprétation des données audio, dépendent de cette valeur.
nChannels	Nombre de voies représentées dans les données audio, par exemple 1 pour la monophonie et 2 pour la stéréophonie.
nSamplesPerSec	Fréquence d'échantillonnage (en échantillons par seconde) à laquelle chaque voie doit être reproduite.
nAvgBytesPerSec	Nombre moyen d'octets de données audio à transférer par seconde. Au moyen de cette valeur, le logiciel de reproduction peut estimer la capacité tampon nécessaire.
nBlockAlign	Alignement en bloc (des octets) des données audio. Le logiciel de reproduction a besoin de traiter un multiple de <nBlockAlign> octets de données à la fois, de sorte que la valeur de ce champ peut être utilisée pour l'alignement du tampon.

Le champ <format-specific-fields> se composent de zéro, un ou plusieurs octets de paramètres. Les paramètres insérés dépendent de la catégorie de format WAVE. On trouvera de plus amples informations dans les paragraphes suivants. Le logiciel de reproduction doit être écrit de façon à permettre (et à ignorer) tous paramètres inconnus apparaissant à la fin du champ <format-specific-fields>.

1.2 Catégories de format WAVE

La catégorie de format d'un fichier WAVE est spécifiée par la valeur du champ <wFormatTag> du fragment 'fmt'. La représentation des données dans le champ <wave-data> et le contenu du champ <format-specific-fields> du fragment 'fmt' dépendent de la catégorie de format.

Les catégories ouvertes (non protégées) actuellement définies dans le format WAVE sont les suivantes:

wFormatTag	Valeur	Catégorie de format
WAVE_FORMAT_PCM	(0x0001)	Format MIC (modulation par impulsions et codage) de Microsoft
WAVE_FORMAT_MPEG	(0x0050)	Audio MPEG-1 (audio seulement)

NOTE – Bien que d'autres formats WAVE aient été déposés par *Microsoft*®, seuls les formats ci-dessus sont actuellement utilisés pour les fichiers BWF. L'article 2 ci-après donne des détails sur le format WAVE de catégorie MIC. L'article 3 donne des informations générales sur d'autres formats WAVE. L'Annexe 2 donne des détails sur le format WAVE de catégorie MPEG. D'autres formats WAVE pourront être définis ultérieurement.

2 Format de modulation par impulsions et codage (MIC)

Si le champ <wFormatTag> du fragment <fmt-ck> est mis à la valeur WAVE_FORMAT_PCM, les données audio se composent d'échantillons modulés en impulsions (MIC), pour lesquelles le champ <format-specific-fields> est défini comme suit:

```
<PCM-format-specific> ->
        struct{
                WORD nBitsPerSample;                    // Longueur d'échantillon
        }
```

Le champ <nBitsPerSample> spécifie le nombre de bits de données utilisés pour représenter chaque échantillon dans chaque voie. S'il y a plusieurs voies, la longueur d'échantillon est la même pour chaque voie.

Pour les données à codage MIC, un champ <nAvgBytesPerSec> du fragment 'fmt' doit être égal à la formule suivante, après avoir été arrondi au plus proche entier:

$$\frac{\text{nChannels} \times \text{nBitsPerSecond} \times \text{nBitsPerSample}}{8}$$

Le champ <nBlockAlign> doit être égal à la formule suivante, après avoir été arrondi au plus proche entier:

$$\frac{\text{nChannels} \times \text{nBitsPerSample}}{8}$$

2.1 Mise en paquet des données pour fichiers WAVE de catégorie MIC

Dans un fichier WAVE monophonique, les échantillons sont enregistrés consécutivement. Pour les fichiers WAVE stéréophoniques, la voie 0 représente la voie de gauche et la voie 1 la voie de droite. La correspondance avec la position des haut-parleurs pour plus de deux voies n'est pas définie actuellement. Dans les fichiers WAVE à voies multiples (multicanaux), les échantillons sont entrelacés.

Les schémas suivants montrent la mise en paquets des données pour des fichiers WAVE mono et stéréo à 8 bits:

Échantillon 1	Échantillon 2	Échantillon 3	Échantillon 4
Voie 0	Voie 0	Voie 0	Voie 0

Mise en paquet des données pour MIC mono 8 bits

Échantillon 1		Échantillon 2	
Voie 0 (gauche)	Voie 1 (droite)	Voie 0 (gauche)	Voie 1 (droite)

Mise en paquet des données pour MIC stéréo 8 bits

Les schémas suivants montrent la mise en paquet des données pour fichiers WAVE mono et stéréo à 16 bits.

Échantillon 1		Échantillon 2	
Voie 0 - octet de poids faible	Voie 0 - octet de poids fort	Voie 0 - octet de poids faible	Voie 0 - octet de poids fort

Mise en paquet des données pour MIC mono à 16 bits

Échantillon 1			
Voie 0 (gauche)	Voie 0 (gauche)	Voie 1 (droite)	Voie 1 (droite)
octet de poids faible	octet de poids fort	octet de poids faible	octet de poids fort

Mise en paquet des données pour MIC stéréo à 16 bits

2.2 Format des données dans les échantillons

Chaque échantillon est contenu dans un nombre entier d'octets i dont la longueur est le plus petit nombre d'octets nécessaires. L'octet de poids faible est enregistré le premier. Les éléments binaires qui représentent l'amplitude d'échantillonnage sont enregistrés aux positions binaires de poids fort du nombre i, les autres bits étant mis à zéro.

Par exemple, si la longueur d'échantillon (indiquée par le champ <nBitsPerSample> est de 12 éléments binaires, chaque échantillon est codé sur un entier de deux octets. Les quatre éléments binaires de poids faible du premier octet (de poids faible) sont mis à zéro. Le format des données et les valeurs extrêmes de diverses longueurs d'échantillons de forme d'onde MIC se présentent comme suit:

Longueur d'échantillon	**Format des données**	**Valeur maximale**	**Valeur minimale**
Un à huit bits	Entier non signé	255 (0xFF)	0
Au moins neuf bits	Entier signé i	Plus grande valeur positive de i	Plus grande valeur négative de i

Par exemple, les valeurs maximale, minimale et médiane des données de forme d'onde MIC codées sur 8 bits et sur 16 bits sont les suivantes:

Format	Valeur maximale	Valeur minimale	Valeur médiane
MIC 8-bits	255 (0xFF)	0	128 (0x80)
MIC 16-bits	32767(0x7FFF)	-32768(-0x8000)	0

2.3 Exemples de fichiers WAVE de type MIC

Exemple de fichier WAVE de type MIC avec fréquence d'échantillonnage de 11,025 kHz, mono, 8 bits par échantillon:

RIFF(‘WAVE’ fmt(1, 1, 11025, 11025, 1, 8)

data(<wave-data>))

Exemple de fichier WAVE de type MIC avec fréquence d'échantillonnage de 22,05 kHz, stéréo, 8 bits par échantillon:

RIFF(‘WAVE’ fmt(1, 2, 22050, 44100, 2, 8)

data(<wave-data>))

Exemple de fichier WAVE de type MIC avec fréquence d'échantillonnage de 44,1 kHz, mono, 20 bits par échantillon:

RIFF(‘WAVE’ INFO(INAM("O Canada"Z))

fmt(1, 1, 44100, 132300, 3, 20)

data(<wave-data>))

2.4 Enregistrement des données de type WAVE

Le champ <**wave-data**> contient les données de forme d'onde sonore. Il est défini comme suit:

<wave-data> -> { <data-ck> }

<data-ck> -> data(<wave-data>)

2.5 Fragment factuel

Le fragment factuel <fact-ck> contient des informations importantes sur le contenu du fichier WAVE. Ce fragment est défini comme suit:

<fact-ck> -> fact(<dwFileSize:DWORD>) // Nombre d'échantillons

Ce fragment n'est pas requis pour les fichiers à modulation MIC.

Le fragment “factuel” sera élargi pour inclure toute autre information requise par de futurs formats WAVE. Les champs ajoutés seront placés après le champ <dwFileSize>. Les applications pourront utiliser le champ de dimension de fragment pour déterminer les champs présents.

2.6 Autres fragments facultatifs

Un certain nombre d'autres fragments sont spécifiés pour usage dans le format WAVE. Les détails de ces fragments sont donnés dans la spécification du format WAVE et dans ses mises à jour ultérieures.

NOTE – Le format WAVE peut prendre en charge d'autres fragments facultatifs qui peuvent être inclus dans des fichiers WAVE pour transporter des informations spécifiques. Comme indiqué dans la note du § 2.1 de l'Annexe 1, ces fragments sont considérés, dans le format BWF, comme étant privés. Ils seront ignorés par les applications qui ne peuvent pas les interpréter.

3 Autres types de fichiers WAVE

Les informations suivantes ont été extraites des normes de données *Microsoft*®, mises à jour au 15 avril 1994. Elles décrivent les extensions nécessaires des fichiers WAVE de base (utilisés pour l'audio à codage MIC) afin de traiter d'autres types de format WAVE.

3.1 Informations générales

Tous les types WAVE nouvellement définis doivent contenir à la fois un fragment factuel et une description de format WAVE étendu, insérée dans le fragment de format <fmt-ck>. Les fichiers WAVE de format RIFF et de type WAVE_FORMAT_PCM n'ont pas besoin de comporter le fragment supplémentaire ni la description de format WAVE étendu.

3.2 Fragment factuel

Ce fragment contient des informations variables selon les fichiers, concernant le contenu du fichier WAVE. Il spécifie actuellement la longueur du fichier, en nombre d'échantillons.

Extension du format WAVE

La structure étendue du format WAVE, ajoutée au fragment <**fmt-ck**>, est utilisée pour définir toutes les données audio de format autre que MIC. Elle est décrite comme suit. La structure étendue du format WAVE général est utilisée pour tous les formats autres que MIC.

```
typedef struct waveformat_extended_tag {
    WORD        wFormatTag;         /* type de format */
    WORD        nChannels;          /* nombre de voies (c'est-à-dire mono, stéréo...) */
    DWORD nSamplesPerSec;           /* fréquence d'échantillonnage */
    DWORD nAvgBytesPerSec;          /* pour l'estimation du tampon */
    WORD        nBlockAlign;        /* longueur de bloc de données */
    WORD        wBitsPerSample;     /* Nombre de bits par échantillon de données mono */
    WORD        cbSize;             /* Comptage d'octets de la forme étendue */
} WAVEFORMATEX;
```

Champ	**Description**
wFormatTag	Ce champ définit le type de fichier WAVE.
nChannels	Nombre de voies représentées dans les données audio: 1 pour la monophonie et 2 pour la stéréophonie.
nSamplesPerSec	Fréquence d'échantillonnage du fichier WAVE. La valeur de ce champ doit être 48000 ou 44100, etc. Cette fréquence est également utilisée par le champ de longueur d'échantillon contenu dans le fragment factuel afin de déterminer la durée des données.
nAvgBytesPerSec	Nombre moyen d'octets de données audio à transférer par seconde. Au moyen de valeur <**nAvgBytesPerSec**>, le logiciel de reproduction peut estimer la capacité tampon nécessaire.
nBlockAlign	Alignement en bloc (des octets) des données audio dans le fragment <**data-ck**>. Le logiciel de reproduction a besoin de traiter un multiple de <**nBlockAlign**> octets de données à la fois, de sorte que la valeur de ce champ peut être utilisée pour l'alignement du tampon.
wBitsPerSample	Nombre de bits par échantillon et par voie. Chaque voie est censée avoir la même résolution de codage des échantillons. Si ce champ n'est pas nécessaire, il doit être mis à zéro.
cbSize	Longueur en octets des informations supplémentaires contenues dans l'en-tête de format WAVE, à l'exclusion de la longueur de la structure d'extension WAVEFORMATEX.

NOTE – Les champs qui suivent le champ <cbSize> contiennent des informations spécifiques qui sont nécessaires pour le format défini dans le champ <wFormatTag>. Tous les formats WAVE qui peuvent être utilisés dans le format BWF seront spécifiés dans des Compléments particuliers de la présente Recommandation.

RÉFÉRENCES

Microsoft® Resource Interchange File Format (RIFF).

Microsoft® Software Developers Kit Multimedia Standards Update, Rev. 3.0, 15 avril 1994.

ANNEXE 2

Spécification du format d'onde de radiodiffusion - Format des fichiers de données audio en radiodiffusion

SPÉCIFICATIONS A UTILISER AVEC LES DONNÉES AUDIO A CODAGE MPEG

1 Introduction

La présente annexe contient la spécification relative à l'utilisation du format BWF afin d'acheminer des signaux audio à codage MPEG seulement. Pour ces signaux, il est nécessaire d'ajouter les informations suivantes aux fragments de base qui sont spécifiés dans le corps de cette Recommandation:

- une extension du fragment de format;
- un fragment factuel;
- un fragment d'extension MPEG.

L'extension du fragment de format et le fragment factuel sont l'un et l'autre spécifiés dans le cadre du format WAVE. Les informations correspondantes sont données dans l'Appendice 1 de l'Annexe 2.

La spécification du fragment d'extension MPEg est donnée dans l'article 2 de l'Annexe 2.

Le corps de la présente Recommandation contient la spécification du fragment d'extension de données audio de radiodiffusion qui est utilisé dans tous les fichiers au format BWF. Les informations relatives au format RIFF de base sont dans l'Appendice 1 de la présente Annexe 2.

2 Audio MPEG

Microsoft® a spécifié la façon dont les données audio MPEG peuvent être organisées en fichiers WAVE. Une extension du fragment de format et un fragment factuel acheminent les informations nécessaires pour spécifier les options de codage MPEG. Les principes généraux sont donnés dans l'Appendice 1 de l'Annexe 1 et les détails dans l'Appendice 1 de l'Annexe 2. Pour la couche 2 du format MPEG, l'on a constaté qu'il fallait acheminer des informations supplémentaires concernant le codage du signal. Ces informations sont insérées dans le fragment d'extension de données audio MPEG <**MPEG Audio Extension**>, mis au point par le groupe d'intérêts pour les données audio de couche 2 MPEG. Ce fragment est spécifié ci-dessous.

2.1 Fragment d'extension de données audio MPEG

Le fragment d'extension de données audio MPEG est défini comme suit:

```
typedef struct {
     DWORD   ckID;            /* (mpeg_extension)ckID='mext' */
     DWORD   ckSize;          /* longueur du fragment d'extension :
                                 cksize=000C*/
     BYTE ckData[ckSize];     /* données du fragment */
}
typedef struct mpeg_audio_extension {
WORD SoundInformation;       /* informations supplémentaires sur le son */
WORD FrameSize;              /* longueur nominale d'une trame */
WORD AncillaryDataLength;    /* longueur des données auxiliaires */
WORD AncillaryDataDef;       /* type de données auxiliaires */
CHAR Reserved [4];           /* champ réservé pour usage futur; forcé à néant */
} MPEG_EXT ;
```

Champ	Description
SoundInformation	Champ de 16 bits donnant des informations supplémentaires sur le fichier audio: Pour la couche II (ou la couche I) MPEG: Bit 0 à 1: Données audio homogènes Bit 0 à 0: Données audio non homogènes Les Bits 1 et 2 sont utilisés pour insérer des informations additionnelles pour fichiers audio homogènes: Bit 1 à 0: Un bit de bourrage est utilisé dans le fichier et peut donc alterner entre 0 ou 1 Bit 1 à 1: Le bit de bourrage est mis à 0 dans tout le fichier Bit 2 à 1: Le fichier contient une séquence de trames dont le bit de bourrage est mis à 0 et dont la fréquence d'échantillonnage est égale à 22,05 ou 44,1 kHz.

NOTE – Un tel fichier n'est pas conforme à la norme MPEG (§ 2.4.2.3, définition du bit de bourrage) mais peut être considéré comme un cas particulier du débit binaire variable (VBR). Il n'est pas nécessaire dávoir un décodeur MPEG pour décoder correctement un tel flux binaire, car la plupart des décodeurs rempliront cette fonction. Le débit sera légèrement inférieur à la valeur indiquée dans l'en-tête.

	Bit 3 à 1: Utilisation d'un format libre Bit 3 à 0: Aucune trame audio à format libre
FrameSize	Nombre d'octets d'une trame nominale, codé sur 16 bits. Ce champ n'a de sens que pour des fichiers homogènes; sinon, il est forcé à 0. Si le bit de bourrage n'est pas utilisé, c'est-à-dire si ce bit reste constant dans toutes les frames du fichier audio, le champ FrameSize contient la même valeur que le champ <nBlockAlign> du fragment de format. Si le bit de bourrage est utilisé et que les données audio se présentent en longueurs variables, le champ FrameSize indique la longueur d'une trame dont le bit de bourrage est mis à 0. La longueur d'une trame dont le bit de bourrage est mis à 1 est d'un octet supplémentaire (de quatre octets supplémentaires pour la couche I), c'est-à-dire <FrameSize+1>. Le fait que le champ <nBlockAlign> soit mis à 1 indique que les trames ont des longueurs variables (FrameSize ou FrameSize+1) avec un bit de bourrage variable.
AncillaryDataLength	Nombre codé sur 16 bits, indiquant le nombre minimal d'octets connus pour les données auxiliaires contenues dans le fichier audio complet. Cette valeur est calculée à partir de la fin de la trame audio.
AncillaryDataDef	Cette valeur de 16 bits spécifie comme suit le contenu des données auxiliaires: Bit 0 mis à 1: L'énergie de la voie gauche est présente dans les données auxiliaires Bit 1 mis à 1: Octet privé, libre pour usage interne dans les données auxiliaires Bit 2 mis à 1: L'énergie de la voie droite est présente dans les données auxiliaires Bit 3 mis à 0: Champ réservé pour usage futur de données ADR Bit 4 mis à 0: Champ réservé pour usage futur de données DAB Bit 5 mis à 0: Champ réservé pour usage futur de données J.52 Bits 6 à 15 mis à 0: Champs réservés pour usage futur

NOTES

– Les éléments présents dans les données auxiliaires suivent le même ordre que les positions binaires du champ AncillaryDataDef. Le premier élément est enregistré à la fin des données auxiliaires, le deuxième juste avant le premier, etc., d'arrière en avant.

– Pour un fichier mono, le bit 2 est toujours mis à 0 et le bit 0 concerne l'énergie de la trame monophonique.

– Pour un fichier stéréo, si le bit 2 est mis à 0 et le bit 0 à 1, l'énergie concerne la valeur maximale des voies gauche et droite.

– L'énergie est enregistrée dans 2 octets et correspond à la valeur absolue de l'échantillon de longueur maximale utilisé pour coder la trame. Il s'agit d'une valeur codée sur 15 bit en format "gros-boutiste" (premier octet supérieur au second).

Reserved	Champ de 4 octets réservé pour usage futur. Ces 4 octets doivent être forcés à néant. Dans tout usage futur, la valeur nulle sera utilisée comme valeur par défaut afin d'assurer la compatibilité.

APPENDICE 1 (DE L'ANNEXE 2)

Format RIFF de fichier audio (.WAV)

Cet appendice spécifie les informations supplémentaires qui sont nécessaires à un fichier WAVE contenant des données audio MPEG.

Les informations contenues dans le présent appendice sont extraites des spécifications du format de fichier RIFF de *Microsoft*®. Elles ne figurent ici qu'à titre documentaire.

Pour de plus amples informations, voir la plus récente version de l'utilitaire *Software Developers Kit* de *Microsoft*®, mis à jour pour les normes multimédias (Rev. 3.0 du 15 avril 1994 ou plus récent).

1 Format des fichiers (audio seulement) MPEG-1

1.1 Fragment factuel

Ce fragment est nécessaire pour tous les formats WAVE autres que WAVE_FORMAT_PCM. On y enregistre des informations variables selon les fichiers au sujet du contenu des données WAVE. Il spécifie actuellement la durée des données en nombre d'échantillons.

NOTE – Voir également l'Appendice 1 de l'Annexe 1, § 2.5.

1.2 En-tête de format WAVE

```
#define WAVE_FORMAT_MPEG      (0x0050)
typedef struct mpeg1waveformat_tag {
      WAVEFORMATEX              wfx;
      WORD    fwHeadLayer;
      DWORD   dwHeadBitrate;
      WORD    fwHeadMode;
      WORD    fwHeadModeExt;
      WORD    wHeadEmphasis;
      WORD    fwHeadFlags;
      DWORD   dwPTSLow;
      DWORD   dwPTSHigh;
} MPEG1WAVEFORMAT;
```

Champ	Description
wFormatTag	Ce champ doit être mis à la valeur WAVE_FORMAT_MPEG *[0x0050]*
nChannels	Nombre de voies représentées dans les données audio: 1 pour la monophonie et 2 pour la stéréophonie.
nSamplesPerSec	Fréquence d'échantillonnage (en Hz) du fichier audio: 32 000 kHz, 44 100 kHz ou 48 000 kHz, etc. On notera cependant que si la fréquence d'échantillonnage des données est variable, ce champ doit être mis à zéro. Il est fortement recommandé d'utiliser une fréquence d'échantillonnage constante pour les applications de bureau.
NAvgBytesPerSec	Débit moyen des données; si le codage à débit binaire variable est utilisé en couche III, la valeur de ce champ peut ne pas désigner un débit conforme MPEG.

NBlockAlign

Alignement en bloc (des octets) des données audio contenues dans le fragment <**data-ck**>. Pour les flux audio qui ont une longueur fixe de trame audio, l'alignement de bloc est égal à la longueur de la trame. Pour les flux dans lesquels la longueur des trames varie, le champ <**nBlockAlign**> doit être mis à 1.

Avec une fréquence d'échantillonnage de 32 ou de 48 kHz, la longueur d'une trame audio MPEG est fonction du débit. Si un flux audio utilise un débit constant, la longueur des trames audio est constante. Les formules suivantes s'appliquent donc:

Couche I: nBlockAlign = 4*(int)(12*BitRate/SamplingFreq)

Couches II et III: nBlockAlign = (int)(144*BitRate/SamplingFreq)

Exemple 1: pour la couche I, avec une fréquence d'échantillonnage de 32 000 kHz et un débit de 256 kbit/s, nBlockAlign=384 octets.

Si un flux audio contient des trames à différents débits, la longueur de ces trames varie à l'intérieur de ce flux. Des longueurs de trame variables se produisent également lorsque l'on utilise une fréquence d'échantillonnage de 44,1 kHz: pour conserver la valeur nominale de ce débit, la longueur de chaque trame audio MPEG est périodiquement augmentée d'un “cran” (4 octets dans la couche I, 1 octet dans les couches II et III) par rapport aux formules indiquées ci-dessus. Dans ces deux cas, la notion d'alignement de bloc est invalide. La valeur du champ **<nBlockAlign>** doit donc être mise à 1, de manière que les applications compatibles MPEG puissent préciser si les données sont alignées en bloc ou non.

NOTE – Il est possible de construire un flux audio possédant des trames audio de longueur constante à 44,1 kHz, en donnant au bit de bourrage contenu dans chaque en-tête de trame audio la même valeur (soit 0 soit 1). Cependant, le débit du flux résultant ne correspondra pas exactement à la valeur nominale indiquée dans l'en-tête de trame et certains décodeurs n'auront peut-être pas la capacité de décoder correctement le flux. Aux fins de la normalisation et de la compatibilité, cette méthode est déconseillée.

WBitsPerSample

Champ non utilisé, mis à zéro.

CbSize

Longueur (en octets) de l'information étendue selon la structure WAWEFORMATEX. Dans le format normalisé WAVE_FORMAT_MPEG, cette longueur est de 22 octets (0x00 16). Si des champs supplémentaires sont ajoutés, cette valeur augmente.

fwHeadLayer

Couche de données audio MPEG, définie par les fanions suivants:

ACM_MPEG_LAYER1 - couche I.

ACM_MPEG_LAYER2 - couche II.

ACM_MPEG_LAYER3 - couche III

Certains flux conformes MPEG peuvent contenir des trames de différentes couches. Dans ce cas, les fanions ci-dessus doivent être mis à zéro ensemble, de manière qu'un pilote puisse déterminer quelles sont les couches présentes dans le flux.

dwHeadBitrate

Débit des données, en bits par seconde. Cette valeur doit correspondre à un débit normalisé selon la spécification MPEG; tous les débits ne sont pas valides pour tous les modes et toutes les couches. Voir les Tableaux 1 et 2. On notera que ce champ enregistre le débit binaire réel et non pas le code contenu dans l'en-tête de trame MPEG. Si le débit est variable, ou s'il s'agit d'un débit non normalisé, ce champ doit être mis à zéro. Il est recommandé d'éviter, chaque fois que possible, le débit variable.

fwHeadMode

Mode de flux, défini par les fanions suivants:

ACM_MPEG_STEREO - stéréophonie.

ACM_MPEG_JOINTSTEREO - stéréophonie mixte (à réduction des sons communs, selon la puissance ou par diaphonie dynamique).

ACM_MPEG_DUALCHANNEL - son bicanal (par exemple, un flux bilingue).

ACM_MPEG_SINGLECHANNEL - son monocanal.

Certains flux conformes MPEG peuvent contenir des trames de différents modes. Dans ce cas, les fanions ci-dessus doivent être mis à zéro ensemble, de façon qu'un pilote puisse préciser quels modes sont présents dans le flux. Cette situation est particulièrement probable en codage stéréophonique mixte car les codeurs peuvent juger utile de commuter dynamiquement entre le mode stéréophonique pur et le mode stéréophonique mixte en fonction des caractéristiques du signal. Dans ce cas, les deux fanions ACM_MPEG_STEREO et ACM_JOINTSTEREO doivent être activés.

fwHeadModeExt

Ce champ contient des paramètres supplémentaires pour le codage en mode stéréophonique mixte; il n'est pas utilisé pour les autres modes. Voir le Tableau 3. Certains flux conformes MPEG peuvent contenir des trames ayant différentes extensions de mode. Dans ce cas, les valeurs indiquées dans le Tableau 3 peuvent être mises à zéro ensemble. On notera que le champ fwHeadModeExt n'est utilisé que pour le codage en mode stéréophonique mixte; il doit être mis à zéro pour les autres modes (monocanal, bicanal, stéréophonie).

En général, les codeurs commutent dynamiquement entre les diverses valeurs d'extension de mode possibles, en fonction des caractéristiques du signal. Pour le codage stéréophonique mixte normal, ce champ doit donc être mis à 0x000f. Si toutefois il est souhaitable de limiter le codeur à un type particulier de codage stéréophonique mixte, ce champ peut être utilisé pour spécifier les types admissibles.

wHeadEmphasis

Ce champ décrit la désaccentuation requise par le décodeur; cela implique que l'accentuation soit appliquée au flux avant son codage. Voir le Tableau 4.

fwHeadFlags

Ce champ active les fanions suivants dans l'en-tête de trame audio:

ACM_MPEG_PRIVATEBIT - active le bit privé.

ACM_MPEG_COPYRIGHT - active le bit de copyright.

ACM_MPEG_ORIGINALHOME - active le bit d'origine.

ACM_MPEG_PROTECTIONBIT - active le bit de protection et insère dans chaque trame un code de protection contre les erreurs de 16 bits.

ACM_MPEG_ID_MPEG1 - met à 1 le bit d'identificateur qui définit le flux comme étant de type audio MPEG-1. *Ce fanion doit toujours être activé explicitement afin d'assurer la compatibilité avec les futures extensions audio MPEG (c'est-à-dire avec MPEG-2).*

Un codeur utilisera la valeur de ces fanions pour activer les bits correspondants dans l'en-tête de chaque trame audio MPEG. Lorsqu'ils décrivent un flux de données codées, ces fanions représentent un OU logique entre les fanions activés dans chaque en-tête de trame. En d'autres termes, si le bit de copyright est activé dans un ou plusieurs en-têtes de trame du flux, le fanion ACM_MPEG_COPYRIGHT sera activé. La valeur de ces fanions n'est donc pas forcément valide pour chaque trame audio.

dwPTSLow

Ce champ (de même que le champ suivant) se compose du pointeur temporel de présentation (PTS) de la première trame du flux audio, extraite de la couche système MPEG. Le champ dwPSTLow contient les 32 bits de poids faible du pointeur PTS de 33 bits. Le pointeur PTS peut être utilisé pour faciliter la réintégration d'un flux audio dans un flux vidéo associé. Si le flux audio n'est pas associé à une couche système, ce champ doit être mis à zéro.

dwPTSHigh

Ce champ (de même que le champ précédent) se compose du pointeur temporel de présentation (PTS) de la première trame du flux audio, extraite de la couche système MPEG. La position binaire de poids faible du champ dwPSTLow contient le bit de poids fort du pointeur PTS de 33 bits. Le pointeur PTS peut être utilisé pour faciliter la réintégration d'un flux audio dans un flux vidéo associé. Si le flux audio n'est pas associé à une couche système, ce champ doit être mis à zéro.

NOTE – Les deux champs précédents peuvent être traités comme un seul entier codé sur 64 bits; facultativement, le champ dwPTSHigh peut être testé en tant que fanion pour déterminer si le bit de poids fort est activé ou non.

TABLEAU 1

Débits admissibles (bit/s)

Code d'en-tête de trame MPEG	Couche I	Couche II	Couche III
’0000’	format libre	format libre	format libre
’0001’	32000	32000	32000
’0010’	64000	48000	40000
’0011’	96000	56000	48000
’0100’	128000	64000	56000
’0101’	160000	80000	64000
’0110’	192000	96000	80000
’0111’	224000	112000	96000
’1000’	256000	128000	112000
’1001’	288000	160000	128000
’1010’	320000	192000	160000
’1011’	352000	224000	192000
’1100’	384000	256000	224000
’1101’	416000	320000	256000
’1110’	448000	384000	320000
’1111’	interdit	interdit	interdit

TABLEAU 2

Combinaisons de débits et de modes admissibles pour la Couche II

Débit (bit/s)	Modes admissibles
32000	monocanal
48000	monocanal
56000	monocanal
64000	tous modes
80000	monocanal
96000	tous modes
112000	tous modes
128000	tous modes
160000	tous modes
192000	tous modes
224000	stéréo, stéréo mixte, bicanal
256000	stéréo, stéréo mixte, bicanal
320000	stéréo, stéréo mixte, bicanal
384000	stéréo, stéréo mixte, bicanal

TABLEAU 3

Extension de mode

fwHeadModeExt	Code d'en-tête de trame MPEG	Couches I et II	Couches III
0x0001	’00’	sous-bandes 4 à 31 en stéréo mixte (en puissance)	pas de codage de stéréo mixte (en puissance) ni de stéréo à signaux *M* et *S* multiplexés
0x0002	'01'	sous-bandes 8 à 31 en stéréo mixte (puissance)	stéréo mixte (puissance)
0x0004	'10'	sous-bandes 12 à 31 en stéréo mixte (puissance)	stéréo M/S
0x0008	'11'	sous-bandes 16 à 31 en stéréo mixte (puissance)	codage de stéréo mixte (puissance) et de stéréo M/S

TABLEAU 4

Champ d’accentuation

wHeadEmphasis	Code d'en-tête de trame MPEG	Désaccentuation requise
1	’00’	pas d’accentuation
2	’01’	accentuation de 50/15 µs
3	’10’	champ réservé
4	'11'	Rec. CCITT J.17

1.3 Fanions utilisés dans les champs de données

fwHeadLayer

Les fanions suivants sont définis pour le champ <fwHeadLayer>. Pour le codage, l'un de ces fanions doit être activé, de façon à indiquer au codeur la couche à utiliser. Pour le décodage, le pilote peut vérifier ces fanions afin de déterminer s'il est en mesure de décoder le flux. On notera qu'un flux conforme MPEG peut utiliser différentes couches dans différentes trames d'un même flux. Plusieurs de ces fanions peuvent donc être activés en même temps.

```
#define ACM_MPEG_LAYER1                 (0x0001)
#define ACM_MPEG_LAYER2                 (0x0002)
#define ACM_MPEG_LAYER3                 (0x0004)
```

fwHeadMode

Les fanions suivants sont définis pour le champ <fwHeadMode>. Pour le codage, l'un de ces fanions doit être activé, de façon à indiquer au codeur la couche [le mode?] à utiliser; pour le codage stéréophonique mixte, les deux fanions ACM_MPEG_STEREO et ACM_MPEG_JOINTSTEREO seront normalement activés, de manière que le codeur puisse n'utiliser le codage stéréophonique mixte que si celui-ci est plus efficace que le codage stéréophonique classique. Pour le décodage, le pilote peut vérifier ces fanions afin de déterminer s'il est en mesure de décoder le flux. On notera qu'un flux conforme MPEG peut utiliser différentes couches dans différentes trames d'un même flux. Plusieurs de ces fanions peuvent donc être activés en même temps.

```
#define ACM_MPEG_STEREO                 (0x0001)
#define ACM_MPEG_JOINTSTEREO            (0x0002)
#define ACM_MPEG_DUALCHANNEL            (0x0004)
#define ACM_MPEG_SINGLECHANNEL          (0x0008)
```

fwHeadModeExt

Le Tableau 3 définit les fanions pour le champ <fwHeadModeExt>. Ce champ n'est utilisé que pour le codage stéréophonique mixte; pour les autres modes de codage, ce champ doit être mis à zéro. Pour le codage stéréophonique mixte, ces fanions indiquent les types de codage stéréophonique mixte qu'un codeur est autorisé à utiliser. Normalement,

un codeur sélectionnera dynamiquement l'extension de mode la mieux appropriée au signal d'entrée; normalement, une application mettra donc ce champ à la valeur 0x000f, de façon que le codeur puisse sélectionner entre toutes les possibilités. Il est toutefois possible de limiter le codeur en éliminant certains de ces fanions. Pour un flux codé, ce champ indique les valeurs du champ MPEG *mode_extension* qui sont présentes dans le flux.

fwHeadFlags

Les fanions suivants sont définis pour le champ <fwHeadFlags>. Ces fanions doivent être activés avant le codage, de façon que les bits appropriés soient activés dans l'en-tête de trame MPEG. Lors de la description d'un flux audio codé MPEG, ces fanions représentent un OU logique entre les bits correspondants de l'en-tête de chaque trame audio. En d'autres termes, si le bit est activé dans une des trames, il l'est également dans le champ <fwHeadFlags>. Si une application enroule un en-tête de fichier WAVE au format RIFF autour d'un flux de données audio binaires précodé MPEG, cette application est chargée d'analyser les éléments de ce flux binaire et d'activer les fanions dans ce champ.

```
#define ACM_MPEG_PRIVATEBIT             (0x0001)
#define ACM_MPEG_COPYRIGHT              (0x0002)
#define ACM_MPEG_ORIGINALHOME           (0x0004)
#define ACM_MPEG_PROTECTIONBIT          (0x0008)
#define ACM_MPEG_ID_MPEG1               (0x0010)
```

1.4 Données audio dans les fichiers MPEG

Le fragment de données se compose d'une séquence de données audio MPEG-1 telle que définie dans l'ISO 11172, Partie 3 (Audio). Cette séquence comporte un flux binaire qui est enregistré dans le fragment de données sous la forme d'un ensemble d'octets. A l'intérieur d'un de ces octets, le bit de poids fort est le premier du flux et le bit de poids faible le dernier. Les données ne sont *pas* inversées dans les octets. Par exemple, les données suivantes sont codées sur les 16 premiers bits (de gauche à droite) d'un en-tête de trame audio normal:

Mot de synchronisation	Identificateur	Couche	Bit de protection	…
111111111111	1	10	1	…

Ces données seront enregistrées dans des octets selon l'ordre suivant:

```
Byte0  Byte1 ...
FF     FD  ...
```

1.4.1 Trames audio MPEG

Une séquence de données audio MPEG se compose d'une série de trames audio dont chacune commence par un en-tête de trame. La plupart des champs contenus dans cet en-tête de trame correspondent à des champs dans la structure MPEG1WAVEFORMAT définie ci-dessus. Pour le codage, ces champs peuvent être activés dans la structure MPEG1WAVEFORMAT et le pilote peut utiliser ces informations pour activer les bits appropriés dans l'en-tête de trame au moment du codage. Pour le décodage, le pilote peut vérifier ces champs afin de déterminer s'il est en mesure de les décoder.

1.4.2 Codage

Un pilote qui code un flux audio MPEG doit lire les champs d'en-tête dans la structure MPEG1WAVEFORMAT puis activer les bits correspondants dans l'en-tête de trame MPEG. Si un pilote a besoin de quelques autres informations, il doit les obtenir soit à partir d'une fenêtre de dialogue de configuration ou au moyen d'une fonction de reprise du pilote. On trouvera de plus amples informations dans le paragraphe ci-après, consacré aux données auxiliaires.

Si un flux audio à précodage MPEG est multiplexé avec un en-tête de format RIFF, il appartient à l'application d'analyser les éléments constituants du flux binaire et d'activer les champs correspondants dans la structure MPEG1WAVEFORMAT. Si la fréquence d'échantillonnage ou la valeur de débit n'est pas constante dans tout le flux de données, le pilote doit mettre à zéro les champs MPEG1WAVEFORMAT correspondants (<nSamplesPerSec> et <dwHeadBitrate>), comme décrit ci-dessus. Si le flux contient des trames appartenant à plusieurs couches, il doit activer les fanions du champ <fwHeadLayer> pour toutes les couches présentes dans le flux. Etant donné que des champs comme <fwHeadFlags> peuvent varier d'une trame à l'autre, il faut prendre des précautions lors de l'activation et du contrôle de ces fanions; en général, une application ne doit pas compter qu'ils seront valides pour chaque trame. Lors de l'activation de ces fanions, il convient d'observer les directives suivantes:

- Le fanion ACM_MPEG_COPYRIGHT doit être activé si l'une des trames du flux contient le bit de copyright activé.
- Le fanion ACM_MPEG_PROTECTIONBIT doit être activé si l'une des trames du flux contient le bit de protection activé.

- Le fanion ACM_MPEG_ORIGINALHOME doit être activé si l'une des trames du flux contient le bit d'origine activé. Ce bit peut être annulé si une copie du flux est faite.
- Le fanion ACM_MPEG_PRIVATEBIT doit être activé si l'une des trames du flux contient le bit privé activé.
- Le fanion ACM_MPEG_ID_MPEG1 doit être activé si l'une des trames du flux contient le bit d'identification activé. Pour les flux MPEG-1 streams, le bit d'identification doit toujours être activé; de futures extensions du format MPEG (comme le format multicanaux MPEG-2) pourront toutefois avoir le bit d'identification annulé.

Si le flux audio MPEG est extrait d'un flux MPEG de couche système, ou si le flux est destiné à être intégré dans la couche système, les champs de pointeur temporel de présentation (PTS) peuvent être utilisés. Le pointeur PTS est un champ de la couche système MPEG qui est utilisé pour la synchronisation des divers champs. Il est codé sur 33 bits et l'en-tête du format RIFF des fichiers WAVE enregistre donc sa valeur en deux fois: le champ <dwPTSLow> contient les 32 bits de poids faible du pointeur PTS et le champ <dwPTSHigh> contient le bit de poids fort. Ces deux champs peuvent être traités ensemble comme un entier de 64 bits; en option, le champ <dwPTSHigh> peut être contrôlé en tant que fanion pour déterminer si le bit de poids fort est activé ou annulé. Lors de l'extraction d'un flux audio d'une couche système, un pilote doit activer les champs de pointeur PTS en fonction des pointeurs PTS de la première trame des données audio. Cette information pourra être utilisée ultérieurement pour réintégrer le flux audio dans la couche système. *Les champs PTS ne doivent pas être utilisés à d'autres fins*. Si le flux audio n'est pas associé à la couche système MPEG, les champs PTS doivent être mis à zéro.

1.4.3 Décodage

Un pilote peut contrôler les champs contenus dans la structure MPEG1WAVEFORMAT afin de déterminer s'il est en mesure de décoder le flux. Il doit cependant tenir compte du fait que certains champs, comme le champ <fwHeadFlags>, peuvent ne pas être constants d'une trame à l'autre du flux binaire. Un pilote ne doit jamais utiliser les champs de la structure MPEG1WAVEFORMAT pour effectuer le décodage proprement dit. Les paramètres de décodage doivent être entièrement extraits du flux de données MPEG.

Un pilote peut contrôler le champ <nSamplesPerSec> afin de déterminer s'il est compatible avec la fréquence d'échantillonnage spécifiée. Si le flux MPEG contient des données à fréquence d'échantillonnage variable, le champ <nSamplesPerSec> doit être mis à zéro. Si le pilote ne peut pas traiter ce type de flux de données, il ne doit pas essayer de les décoder mais doit immédiatement arrêter de fonctionner.

1.5 Données auxiliaires

Les données audio contenues dans une trame audio MPEG ne remplissent pas toujours la totalité de la trame. Les données restantes sont dites *auxiliaires*: elles peuvent avoir tout format désiré et peuvent être utilisées pour acheminer des informations supplémentaires d'un type quelconque. Si un pilote souhaite prendre en charge les données auxiliaires, il doit avoir la capacité d'acheminer ces données en provenance et à destination de l'application appelante. Le pilote peut, à cette fin, faire appel à une fonction de reprise. En principe, le pilote peut appeler une fonction de reprise spécifiée chaque fois qu'il est en possession de données auxiliaires à transmettre à l'application (c'est-à-dire à décoder) ou chaque fois qu'il a besoin d'autres données auxiliaires (à coder).

Les pilotes doivent tenir compte du fait que toutes les applications ne seront pas disposées à traiter les données auxiliaires. Un pilote ne doit donc fournir ce service que lorsque l'application lui en fait la demande expresse. Le pilote peut définir un message particulier qui active et désactive la fonction de reprise. Des messages distincts peuvent être définis pour les opérations de codage et de décodage, afin d'augmenter la flexibilité.

On notera que cette méthode n'est pas forcément appropriée à tous les pilotes ou à toutes les applications; elle n'est présentée qu'à titre d'illustration de la façon dont des données auxiliaires peuvent être prises en charge.

NOTE – On trouvera de plus amples informations sur les données auxiliaires dans le fragment <**MPEG_Audio_Extension**> qu'il y a lieu d'utiliser pour les fichiers MPEG conformes au format BWF. Voir l'article 2 de l'Annexe 2.

RÉFÉRENCES

Microsoft® Resource Interchange File Format, RIFF.

Microsoft® Software Developers Kit Multimedia Standards Update, Rev. 3.0, 15 avril 1994.

ISO/CEI 11173-3: MPEG 1.

ISO/CEI 13818-3: MPEG 2.

NOTE - Les documents d'origine *Microsoft*® sont disponibles à l'adresse Internet suivante: http://www.microsoft.com.