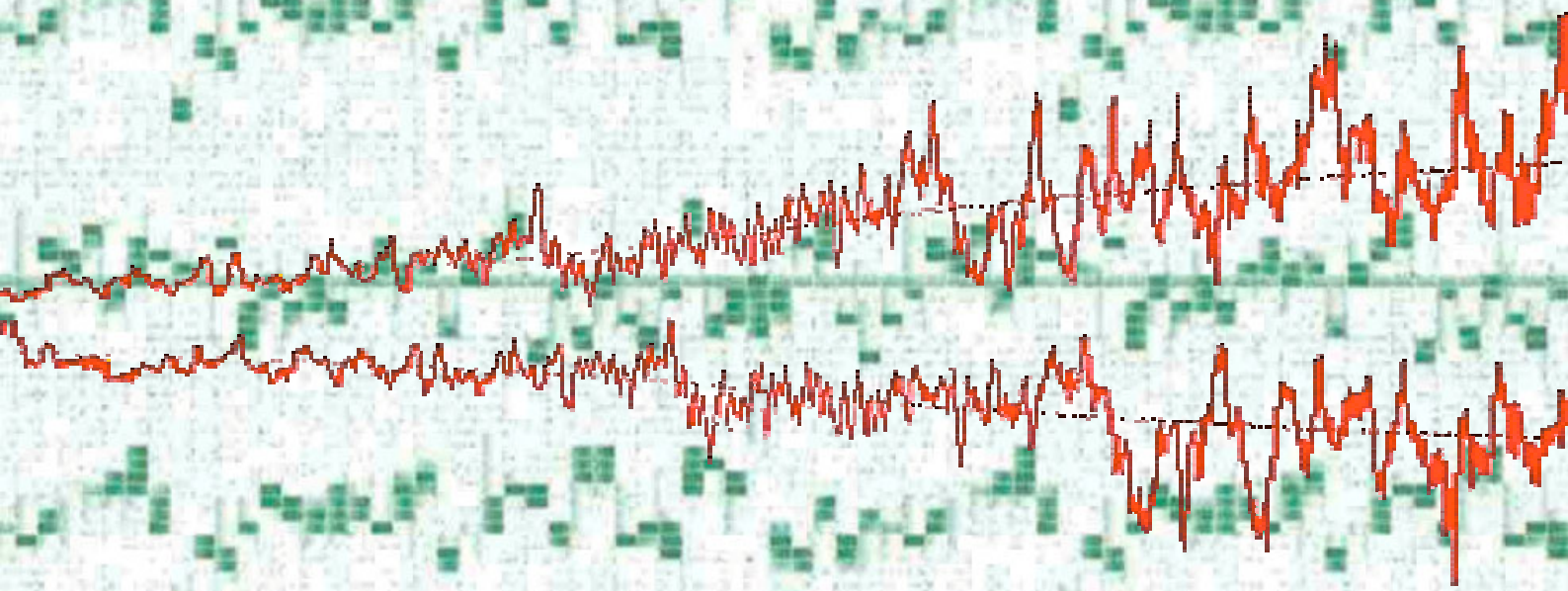
Ingeniería de Ondas

Formatos de Audio Digital



E.T.S. Ingenieros de Telecomunicación
Universidad de Valladolid

Alberto López Martín
alopez@ribera.tel.uva.es

Índice

1. Introducción al audio digital

El audio digital es la representación de señales sonoras mediante un conjunto de datos binarios. Un sistema completo de audio digital comienza habitualmente con un transceptor (micrófono) que convierte la onda de presión que representa el sonido a una señal eléctrica analógica.

Esta señal analógica atraviesa un sistema de procesado analógico de señal, en el que se puede realizar limitaciones en frecuencia, ecualización, amplificación y otros procesos como el de compansión. La ecualización tiene como objetivo contrarrestar la particular respuesta en frecuencia del transceptor utilizado de forma que la señal analógica se asemeje mucho más a la señal audio originaria.

Tras el procesado analógico la señal se muestrea, se cuantifica y se codifica. El muestreo toma un número discreto de valores de la señal analógica por segundo (tasa de muestreo) y la cuantificación asigna valores analógicos discretos a esas muestras, lo que supone una pérdida de información (la señal ya no es la misma que la original). La codificación asigna una secuencia de bits a cada valor analógico discreto. La longitud de la secuencia de bits es función del número de niveles analógicos empleados en la cuantificación. La tasa de muestreo y el número de bits por muestra son dos de los parámetros fundamentales a elegir cuando se quiere procesar digitalmente una determinada señal de audio.

Los formatos de audio digital tratan de representar ese conjunto de muestras digitales (o una modificación) de las mismas de forma eficiente, tal que se optimice en función de la aplicación, o bien el volumen de los datos a almacenar o bien la capacidad de procesamiento necesaria para obtener las muestras de partida. En este sentido hay un formato de audio muy extendido que no se considera de audio digital: el formato MIDI. MIDI no parte de muestras digitales del sonido, sino que almacena la descripción musical del sonido, siendo una representación de la partitura de los mismos.

El sistema de audio digital suele terminar con el proceso inverso al descrito. De la representación digital almacenada se obtienen el conjunto de muestras que representan. Estas muestras pasan por un proceso de conversión digital-analógica proporcionando una señal analógica que tras un procesado (filtrado, amplificación, ecualización, etc.) inciden sobre el transceptor de salida (altavoz) que convierte la señal eléctrica a una onda de presión que representa el sonido.

2. Parámetros fundamentales del audio digital

Los parámetros básicos para describir la secuencia de muestras que representa el sonido son:

- El número de canales: 1 para mono, 2 para estéreo, 4 para el sonido cuadrafónico, etc.
- Tasa de muestreo: El número de muestras tomadas por segundo en cada canal.
- Número de bits por muestra: Habitualmente 8 o 16 bits.

Como regla general, las muestras de audio multicanal suelen organizarse en tramas. Una trama es una secuencia de tantas muestras como canales, correspondiendo cada una a un canal. En este sentido el número de muestras por segundo coincide con el número de tramas por segundo. En estéreo, el canal izquierdo suele ser el primero.

Como se verá más adelante, el sonido telefónico pasa por un proceso de compansión (p.ej. compresión logarítmica) y se codifica en secuencias de 8 bits. Sin embargo, los datos almacenados se corresponden a un rango dinámico lineal de 14 bits, por lo que hay cierta ambigüedad en el número de bits por muestra.

3. Técnicas de compresión

Las técnicas de compresión son objeto de otro de los trabajos de la asignatura, sin embargo están muy relacionadas con los formatos de audio digital por lo que las trataré de forma muy general y breve.

Las técnicas de compresión son la herramienta fundamental de la que se dispone para alcanzar el compromiso adecuado entre capacidad de almacenamiento y de procesamiento requeridas.

Las técnicas de compresión más elaboradas proporcionan una reducción muy importante de la capacidad de almacenamiento, pero requieren también de un importante procesado tanto para compresión como para la descompresión (sobre todo en la compresión). Las técnicas más simples ofrecen reducciones moderadas con poco procesamiento. Las características del sistema digital implicado y la aplicación determinarán el compromiso entre estos factores y permiten seleccionar las técnicas de compresión adecuadas. Las técnicas más avanzadas analizan la respuesta del oído a la señal y simplifican aquellos elementos irrelevantes para la sensación sonora, consiguiendo tasas de compresión mucho mayores.

Algunas técnicas de compresión relevantes:

— **ADPCM** (*Adaptative Differential Pulse Code Modulation*). Se trata de una codificación diferencial: en lugar de representar las muestras de la señal, se almacenan la diferencia entre muestras consecutivas, que para señales audio, suele ser pequeña. ADPCM se articula en los estándares CCITT G.721, CCITT G.723 y en el CCITT G.726, que reemplazó a los dos anteriores definiendo estándares para 16, 24, 32 y 40 kbits por segundo (que corresponden a tamaños de muestra de 2, 3, 4 y 5 bits respectivamente).

— **LPC-10E** (*Linear Predictive Coder*). Este algoritmo hace corresponder la señal audio con un modelo lineal simple y obtiene los parámetros que mejor ajustan el modelo a la señal. La señal generada es poco fiel a la original. Se utiliza en algunos servicios de voz.

— **CELP** (*Code Excited Linear Prediction*). Es similar a LPC-10E, pero además de asignar los parámetros del modelo, tiene en cuenta el error entre la señal original y la aproximada, creando una tabla de errores. La señal se compone de los parámetros del modelo más el índice del error en cada muestra. La tabla es común a codificador y decodificador.

— **GSM 06.10**. Es una modificación de LPC denominada RPE-LPC (*Regular Pulse Excited – Linear Predictive Coder*). La compresión es muy elevada pero requiere también de mucho procesamiento.

— **MPEG**. Es un estándar tanto para audio como para vídeo. Consigue alta compresión en los datos y requiere de mucha potencia de cálculo, sobre todo en la codificación. Hay definidos tres *layers* para las versiónes MPEG-1 y MPEG-2:
  - **Layer I**: desde 32 a 448 kbps
  - **Layer II**: desde 32 a 384 kbps
  - **Layer III**: desde 32 a 320 kbps

4. Calidad del audio digital

La calidad del audio digital depende fuertemente de los parámetros con los que esa señal de sonido ha sido adquirida, pero no son los únicos parámetros importantes para determinar la calidad.

Una forma de estimar la calidad del sonido digital es analizar la señal diferencia entre el sonido original y el sonido reproducido a partir de su representación digital. Según esta estrategia podemos hablar de una relación señal a ruido concreta. Para los sistemas de audio que realicen compresiones digitales tipo *lossless*, esta medida va ha estar determinada por el número de bits por muestra y la tasa de muestreo.

El número de bits por muestra determina un número de niveles de cuantificación y éstos una relación señal a ruido de pico de portadora que depende de forma cuadrática del número de bits por muestra para el caso de la cuantificación uniforme. La tasa de muestreo establece una cota superior para las componentes espectrales que pueden representarse, pudiendo aparecer distorsión lineal en la señal de salida y *aliasing* (o solapamiento de espectros) si el filtrado de la señal no es el adecuado.

Para los sistemas digitales con otro tipo de compresión la relación señal a ruido puede indicar valores muy pequeños aunque las señales sean idénticas para el oído humano.

La razón es que la relación señal a ruido no es un buen parámetro de medida de la calidad de sonido debido a que la calidad que percibe el oyente está determinada por la respuesta del oído humano a las ondas sonoras, que no percibe muchas de las posibles diferencias. Lógicamente si las señales son muy parecidas, el oído no las podrá diferenciar, pero también pueden ser muy distintas y ser percibidas como la señal original. Por lo tanto parece más apropiada la evaluación de la calidad de un sistema digital mediante parámetros de sensibilidad del oído humano y pruebas específicas con oyentes especializados.

Es en este sentido en el que se evalúa la calidad de los sistemas de audio digital en la actualidad. Tanto MPEG como Dolby Digital (AC-3), que establecen compresiones perceptivas, realizan bancos de pruebas para estimar la calidad de las codificaciones.

5. Formatos de fichero

Los formatos de fichero indican la estructura con la que el audio es almacenado. Desde un punto de vista histórico, en los comienzos del audio digital aparecieron multiplicidad de formatos de audio y cada sistema determinaba el formato que utilizaba. Con el tiempo el conjunto de formatos usados se redujo mediante la aparición de formatos cada vez más flexibles y eficientes, llegando a la

situación actual en la que hay unos pocos usados de forma masiva, mientras que el resto tienen usos muy reducidos.

Los formatos de fichero no tienen por qué coincidir con las características del reproductor. En general un mismo formato de fichero permite contener diversas codificaciones, tasas de muestreo, etc. En este sentido se distingue entre dos grupos de formatos de ficheros de audio:

- Formatos autodescriptivos: Contienen de forma explícita los parámetros del dispositivo y la codificación en algún punto del fichero.
- Formatos sin cabecera o tipo raw: Los parámetros del dispositivo y codificación empleada son fijos.

Formatos de fichero autodescriptivos

Como tales, suelen permitir la elección entre varias codificaciones, de entre las cuales se especifica la utilizada en una cabecera.

La cabecera suele comenzar por lo que se conoce como un número mágico, que no es más que un valor fijo que permite identificar el fichero como un fichero del formato buscado. Esta cabecera suele contener la tasa de muestreo, el número de bits por muestra, si las muestras tienen signo o no, si se colocan *little-endian* (último el LSB) o *big-endian* (último el MSB), … y otro tipo de información como descripción del sonido que contiene o notas de *copyright*.

Existe un subconjunto de estos ficheros formados por aquellos que en lugar de tener una cabecera con la información de codificación, organizan el fichero en bloques de datos y bloques de información de codificación, intercalando unos y otros. Estos ficheros permiten la utilización de diversas codificaciones de datos a lo largo de un mismo fichero.

La siguiente tabla muestra una relación de algunos de los formatos de fichero de audio autodescriptivos más habituales e indica algunos de los parámetros que permiten modificar.

EXTENSIÓN	**NOMBRE**	**ORIGEN**	**PARÁMETROS MODIFICABLES**				**COMENTARIOS**
			Tasa	*Canales*	*Codific*	*Otra info*	
.au, .snd		NeXT, SUN	✓	✓	✓	✓(1)	
.aif, .aiff	AIFF	Apple, SGI	✓	✓	✓(2)	✓	
.aif, .aiff	AIFC	Apple, SGI	✓	✓	✓(2)	✓	AIFF con compresión
.iff	IFF/8SVX	AMIGA	✓	✓		✓	Info. de instrumento, 8 bits
.mp2, .mp3	MPEG	MPEG	✓	✓	✓		
.ra	Real Audio	Real Networks	✓	✓	✓		
.sf	IRCAM		✓	✓	✓	✓	
.smp		Turtle Beach					16 bits/1 canal, bucles
.voc		SoundBlaster	✓				8 bits/1canal, puede detectar el silencio
.wav	WAVE	Microsoft	✓	✓	✓(2)	✓	
.wve		Psion					8k sps, 1 canal, ley-A, 8 bits
(ninguna)	HCOM	Mac	✓				8 bits/1 canal, comp. Huffman

(1) Una cadena de información
(2) Sólo longitud de muestra

Formatos de fichero sin cabecera o tipo "raw"

Estos formatos definen un único esquema de codificación y no permiten la variación de los parámetros salvo, en algunos casos, la tasa de muestreo. De hecho, muchas veces no se puede conocer de ninguna forma la tasa de muestreo empleando a menos que se escuche el sonido.

Estos formatos son menos importantes que los autodescriptivos, por ser menos flexibles. Hoy en día están prácticamente en desuso, aunque en el pasado fueron los primeros en aparecer.

EXTENSIÓN	ORIGEN	CARACTERÍSTICAS
.snd, .fssd	Mac, PC	Tasa variable, 1 canal, 8 bits
.ul	US Telephony	8ksps, 1 canal, 8 bits, Ley-μ
.snd	Amiga	Tasa variable, 1 canal, 8 bits con signo

6. Formato AU

En un formato de fichero muy asociado a máquinas Sun y Next. Su estructura es muy sencilla, las razones de compresión que puede llegar a ofrecer son pequeñas y destaca sobre todo su soporte de longitudes de muestras muy altas comparadas con otros formatos (32 y 64 bits). Se compone de tres secciones:

- Una cabecera, en la que describe la codificación de audio utilizada
- Un campo de longitud variable para almacenar otro tipo de información como texto en formato ASCII
- El conjunto de los datos de audio

Cabecera AU

Se compone de seis campos de 32 bits con la siguiente estructura:

```
struct {
      long magic;
      long offset;
      long size;
      long encoding;
      long samplerate;
      long channels;
}AUheader;
```

magic Es el número 0x2E736E64, que representa en ASCII a la cadena ".snd". Este número permite identificar a un fichero como de formato AU.

offset Indica el punto de comienzo de la zona de datos, expresado en bytes. Su valor mínimo es 24, que corresponde a cuando no existen datos adicionales entre cabecera y datos.

size Expresa el tamaño de la región de datos, que si no se conoce debe ser el valor 0xFFFFFFFF.

encoding	Especifica la codificación empleada para el almacenamiento de las muestras de sonido. Algunos de los posibles valores son: 1 – 8 bits Ley-μ RDSI 2 – 8 bits PCM lineal 3 – 16 bits PCM lineal 4 – 24 bits PCM lineal 5 – 32 bits PCM lineal 6 – 32 bits IEEE punto flotante 7 – 64 bits IEEE punto flotante 23 – 8 bits Ley-μ comprimido según el estándar CCITT G.721 (ADPCM) para codificación de voz. Los valores posibles son muchos, los 256 menores están reservados por Next. El resto está disponible para formatos específicos de la aplicación.
samplerate	Tasa de muestreo en muestras por segundo (sps).
channels	Número de canales. Los datos correspondientes a codificaciones multicanal se organizan en tramas de muestra [ver Campo de datos].

Campo de información

Tras la cabecera AU se puede colocar un campo de información de propósito y formato libre. La longitud de este campo está determinado por el campo **offset** de la cabecera. La cabecera tiene longitud fija, de 24 bytes, por lo que este campo tiene una longitud de **offset** – 24 bytes.

Sus usos fundamentales son la inclusión de información de *copyright* en el fichero y la descripción del mismo utilizando caracteres ASCII imprimibles.

Campo de datos

El campo de datos comienza en la posición indicada por **offset** y puede tener longitud no definida.

En configuraciones de formato multicanal las tramas se agrupan en *tramas de muestra*, tal que el campo de datos es una sucesión de tramas de muestra. Una trama de muestra contiene tantas muestras como número de canales, y cada una de las muestras corresponde a un canal diferente. Para la reproducción del fichero será necesario que se obtenga del mismo una tasa de **samplerate** tramas de muestra por segundo.

7. Formatos AIFF y AIFC

El formato AIFF (*Audio Interchange File Format*) está muy extendido en plataformas Apple. Se fundamenta en el formato IFF de Electronic Arts, que permite almacenar la información en segmentos (o *chunks*).

El formato AIFC es una extensión que permite la compresión de los datos de audio.

Cabecera IFF

Al tratarse de un fichero de formato IFF, debe contener al comienzo una cabecera con un primer campo de 4 bytes que contiene la palabra "FORM", un segundo campo de 4 bytes que indica la longitud del resto del fichero. Por último para identificar el fichero IFF como contenedor de audio AIFF, los 4 bytes siguientes a la cabecera deben contener la palabra "AIFF":

```
struct {
   char  id[4];
   DWORD len;
} iff_hdr;

char aiff_id[4];
```

Chunks AIFF

A partir de la cabecera y la identificación el resto del fichero se compone de una secuencia de segmentos. Cada segmento se compone a su vez de una cabecera de segmento compuesta por 4 bytes de identificación y 4 bytes de longitud del campo de datos. Esta longitud no incluye ni la cabecera ni el posible byte que pueda haber para hacer que su longitud total sea par.

El orden de estos segmentos es irrelevante. Sólo existe un segmento obligatorio denominado Segmento Común ("**COMM**") y en el caso de que la forma de onda tenga longitud mayor que cero, también es obligatoria la existencia del segmento Datos de Sonido ("**SSND**").

El resto de segmentos son opcionales y los programas de reproducción podrán ignorarlos selectivamente. Sin embargo a la hora de copiar el fichero se deben copiar la totalidad de los segmentos incluidos los que son ignorados en la reproducción.

Generalidades del formato AIFF

La ordenación de los bytes en formato AIFF es de tipo *big-endian* como en el microprocesador 68000 de Motorola.

Las muestras de la señal se almacenan en el menor número entero bytes, rellenando los bits sobrantes con ceros. En cada muestra los bits de información se sitúan en las posiciones de mayor peso, quedando el relleno de 0's en las posiciones menos significativas.

Las reproducciones multicanal se organizan de la siguiente forma: los muestras se agrupan en *tramas de muestra*, que son un conjunto de muestras, cada una de las cuales corresponde a un canal distinto. Está definido el siguiente orden para las siguientes situaciones:

- **Estéreo**: Izquierdo – Derecho
- **Tres canales**: Izquierdo – Derecho – Central
- **Cuadrafónico**: Delantero Izquierdo – Delantero Derecho – Trasero Izquierdo – Trasero Derecho
- **Cuatro canales**: Izquierdo – Central – Derecho – Entorno
- **Seis canales**: Central Izquierdo – Izquierdo – Central – Central Derecho – Derecho – Entorno

Las muestras pertenecientes a una trama de muestra se empaquetan una tras otra, sin rellenos, al igual que las tramas de muestra entre sí.

Segmento Común (Common Chunk)

Describe parámetros fundamentales de la forma de onda almacenada como la tasa de muestreo, el número de bits por muestra y el número de canales. Se identifica por la secuencia "**COMM**".

```
struct {
      short numChannels;
      unsigned long numSampleFrames;
      short sampleSize;
      extended[1] sampleRate;
}CommonChunk;
```

numChannels	Especifica el número de canales en el segmento de datos.
numSampleFrames	Indica el número de cuadros que contiene el segmento de datos. El número total de muestras es este valor multiplicado por el número de canales.
sampleSize	Especifica el número de bits por muestra. Puede tomar un número entre 1 y 32.
sampleRate	Indica la tasa a la que las muestras se deben reproducir.

Segmento de datos de sonido (Sound Data chunk)

Contiene todos las muestras, las de todos los canales de sonido. Su identificativo es "**SSND**". Es un segmento obligatorio salvo que el campo **numSampleFrames** del segmento común valga cero.

```
struct {
      unsigned long offset;
      unsigned long blockSize;
      unsigned char WaveformData[];
} SoundDataChunk;
```

offset	Indica el comienzo de la primera trama de muestras. En la mayoría de los casos no se empleará y contendrá un valor cero. Su utilización está asociada a la alineación en bloques.
blockSize	Tamaño de bloque. Es el número de bytes de los bloques a los que la forma de onda se alinea.
WaveformData	Los datos de la forma de onda.

[1] El tipo de dato **extended** hace referencia a un número en coma flotante de 80 bits definido en el estándar IEEE 754.

Alineación en bloques de los datos

La alineación de bloques es una necesidad que presentan algunas aplicaciones para poder realizar una reproducción y grabación en tiempo real. Por ello se permite dividir la información en bloques de **blockSizeBytes** y alinearlos mediante offset. Las aplicaciones que no requieran de esta ordenación, no deben usarla y si operan con un fichero con esta ordenación es conveniente que la mantengan, aunque no es obligatorio.

Segmento Marker

Sirve para indicar determinadas posiciones de la forma de onda contenida en el fichero, para cualquier propósito. Las marcas definidas con este segmento, que se identifica por "**MARK**", son utilizadas por otros como referencias.

```
struct {
      unsigned short numMarkers;
      Marker Markers[];
} MarkerChunk;

struct {
      short id;
      unsigned long position;
      pstring markerName;
} Marker;
```

numMarkers	Número de marcas definidas en el segmento.
id	Es un número que identifica de forma única a la marca dentro del fichero AIFF. Debe ser un número mayor que cero.
position	Indica la posición de la marca. En AIFF las marcas se sitúan entre tramas de muestras, considerando que una marca antes de la primera trama de muestra tiene posición cero.
markerName	Es una cadena tipo PASCAL que contiene la descripción de la marca. Estas cadenas tienen como primer elemento la longitud de la cadena, a la que siguen los caracteres.

Segmento de instrumento

Define unos parámetros básicos que un instrumento, como uno MIDI, puede utilizar para reproducir la forma de onda. Se identifica por la cadena "**INST**".

```
struct {
      char baseNote;
      char detune;
      char lowNote;
      char highNote;
      char lowVelocity;
      char highVelocity;
      short gain;
```

```
        Loop sustainLoop;
        Loop releaseLoop;
} InstrumentChunk;
```

baseNote	Indica el número de la nota a la que el instrumento reproduce la forma de onda sin modificación de tono. (Esto es, a la misma tasa de muestras a la que fue creado). Las unidades son notas MIDI, de 0 a 127.
detune	Determina cuanto debe ser modificado el tono cuando el sonido se reproduce. Se especifica en centésimas de semitono, pudiendo variar entre –50 y +50. Los números negativos indican que el tono debe ser bajado.
lowNote y **highNote**	Especifican el rango de notas recomendado en un teclado para la reproducción de la forma de onda. La forma de onda se debe reproducir si al teclado se le pide una nota dentro de este rango, extremos incluidos. El valor de **baseNote** no tiene por qué estar en este rango.
lowVelocity y **highVelocity**	Especifica el rango de velocidades a las que se deben reproducir las notas. Aquellas notas que indiquen una velocidad dentro de este rango, extremos incluidos, deben ser reproducidas. Las unidades son valores MIDI: 1 (la más lenta) y 127 (la más rápida).
gain	Especifica la ganancia que se debe aplicar al sonido cuando se reproduce. Se mide en dB, con 6 dB se multiplican las muestras por 2 y con –6 dB se dividen por 2.
sustainLoop	Especifica un bucle a ser reproducido cuando el instrumento sostiene un sonido.
releaseLoop	Especifica un bucle que se reproduce cuando el instrumento está terminando la reproducción de un sonido.

Los bucles utilizados en este segmento se organizan según la siguiente estructura:

```
struct {
        short PlayMode;
        short beginLoop;
        short endLoop;
} Loop;
```

PlayMode	Especifica el modo de reproducir el bucle. Hay tres formas: ▪ **NoLooping**: No realiza bucles ▪ **ForwardLooping**: Desde el inicio al final ▪ **ForwardBackwardLooping**: Del inicio al final y del final al inicio.
beginLoop	Identifica la marca de comienzo de la forma de onda a reproducir.
endLoop	Identifica la marca de final de la forma de onda a reproducir.

Segmento de datos MIDI

Como su nombre indica permite almacenar datos MIDI en el fichero AIFF. Se identifica por la cadena "**MIDI**". Consta de un único parámetro que es una secuencia de bytes que contiene los datos MIDI.

Su presencia es opcional y puede haber más de uno en un fichero AIFF. En el caso de disponer de información relativa a varios instrumentos, se recomienda que exista un segmento "**MIDI**" por instrumento en lugar de un único segmento para todos.

Su propósito fundamental es contener los denominados *MIDI System Exclusive Messages* asociados sobre todo a parámetros MIDI que aparezcan después de la especificación de AIFF y que por tanto no están recogidos en el segmento de instrumento o que recogen capacidades exclusivas de algún instrumento que los demás no pueden interpretar.

Segmento de grabación de audio

Se identifica por "**AESD**". Contiene una tabla de información descrita en el "*AES Recommended Practice for Digital Audio Engineering - Serial Transmission Format for Linearly Represented Digital Audio Data*".

Su presencia es opcional y no puede haber más de uno en un fichero AIFF. Esa información se incluye en el formato AIFF por conveniencia, al describir parámetros útiles para la grabación del sonido.

Segmento específico de aplicación

Permite introducir datos para cualquier fin tanto a desarrolladores como a creadores de aplicaciones. Se identifica mediante la cadena "**APPL**".

```
struct {
      char applicationSignature[4];
      char data[];
} ApplicationSpecificChunk;
```

applicationSignature	Permite distinguir entre diversos segmentos específicos de aplicaciones.
data	Datos relacionados con la aplicación.

Segmento de comentarios

El formato IFF define un segmento para la introducción de comentarios que también puede ser usado. El segmento de comentarios AIFF, identificado por "**COMT**" tiene una estructura más elaborada, su presencia es opcional pero no debe aparecer más de un segmento por fichero.

```
struct {
      unsigned long timeStamp;
      short marker;
      unsigned short count;
      char text[];
} Comment;
```

timeStamp	Especifica el momento en el que el comentario se creó. En plataformas Amiga se mide en segundos desde el 1 de Enero de 1978; en Mac, desde el 1 de Enero de 1904.
marker	Su su valor es distinto de cero, identifica un marca, permitiendo hacer comentarios sobre marcas en la forma de onda. Si vale cero, el comentario no se asocia a ninguna marca.
count	Indica la longitud del texto que forma el comentario. Al ser de 16 bits permite comentario mucho mayores a los que se podrían realizar con una cadena tipo PASCAL.
text	Cadena de texto que constituye el comentario.

Segmentos de texto: nombre, autor, copyright y anotación

Este conjunto de segmentos se identifican, respectivamente, por "**NAME**", "**AUTH**", "**(c)** " y "**ANNO**". Todos tienen la misma estructura. Sólo contienen un elemento, que es la cadena de caracteres que describe lo que el segmento indica en cada caso.

8. Formatos MPEG

MPEG es un grupo de trabajo de un subcomité de ISO/IEC (*International Organization for Standarization / International Electrotechnical Commission*) encargada del desarrollo internacional de estándares para la compresión, decompresión, procesado y representación codificada de vídeo, audio y su combinación.

Los estándares MPEG se desarrollan en fases numeradas, por lo que la especificación MPEG-2 no es una sustitución de MPEG-1 sino una ampliación o complemento del mismo. Lo que se conoce por "*layers*", son especificaciones de una familia de algoritmos de codificación para un mismo estándar.

Para las señales de audio MPEG ha definido los estándares MPEG-1, MPEG-2, MPEG-4 y MPEG-7 que proporcionan:

NOMBRE	**ESTÁNDAR**	**CARACTERÍSTICAS**
MPEG-1	ISO/IEC 11172-3	Proporciona codificación de un canal (mono) o dos canales (estéreo o mono dual) con tasas de muestreo de 32, 44.1 y 48 ksps. Las tasas de bits (*bitrates*) predefinidas son: ▪ **Layer I**: De 32 a 448 kbps ▪ **Layer II**: De 32 a 384 kbps ▪ **Layer III**: De 32 a 320 kbps
MPEG-2 BC	ISO/IEC 13818-3	Proporciona: ▪ Una extensión multicanal compatible con MPEG-1 (*Backwards Compatible*: BC). Permite hasta 5 canales principales y además un canal de mejora de bajas frecuencias. Las tasas binarias se extienden hasta valores próximos a 1 Mbps. ▪ Una extensión para tasas de muestreo menores. Ofrece tasas a 16, 22.05 y 24 ksps para *bitrates* de 32 a 256 kbps para el *Layer I* y de 8 a 160 kbps para los *Layers II* y *III*.
MPEG-2 AAC	ISO/IEC 13818-7	Es un estándar de codificación audio de muy alta calidad de hasta 48 canales con tasas de muestreo desde 8 a 96 ksps con capacidades multicanal, multilenguaje y multiprograma. Trabaja a *bitrates* desde 8 kbps para señal monofónica de voz hasta más de 160 kbps/canal para codificación de muy alta calidad.
MPEG-4	ISO/IEC 14496-3	Proporciona: ▪ Codificación y composición de objetos audio tanto naturales como sintetizados. ▪ Escalabilidad en el *bitrate*. ▪ Escalabilidad en la complejidad de los codificadores y decodificadores. ▪ Audio Estructurado: Lenguaje universal para la síntesis de sonido. ▪ TTSI: Un interfaz para la conversión de texto a voz.
MPEG-7	ISO/IEC 15938	Proporciona: ▪ Descripciones estandarizadas y esquemas de descripción de estructuras de audio y contenido audio. ▪ Un lenguaje para esas descripciones y esquemas de descripciones.

Existe además la versión denominada MPEG-2.5, que no es un estándar oficial, sino propietario del *Fraunhofer Institute for Integrated Circuits*, Alemania. Es una extensión del MPEG-1/2 *Layer III* que permite codificar a frecuencias de muestreo aún menores (8, 11.025 y 12 ksps).

El estándar MPEG-7 sirve para la incorporación de metainformación con la información de audio, y por tanto, está fuera del alcance de este trabajo. Nos centraremos sobre todo en los estándares MPEG-1 y MPEG-2 que sí definen formas concretas en las que se estructura el audio digital.

MPEG-1 Audio

El estándar MPEG-1 se terminó en 1992 y constituyó el estándar internacional ISO/IEC 11172-3, publicado en 1993.

Estandariza tres esquemas distintos para la codificación de ondas de sonido denominados *Layers I, II* y *III*. No estandariza el codificador sino el tipo de información que éste debe producir y cómo un decodificador debe dividir, descomprimir y sintetizar esta información para obtener el sonido codificado. Puede combinarse la secuencia de bits (*bitstream*) audio con vídeo en lo que se conoce como un *bitstrem* de sistemas MPEG-1.

Codificación

La codificación MPEG-1 es general para cualquier señal de audio. La compresión que realiza no es de tipo *lossless* sino perceptiva. La codificación *lossless* aprovecha la redundancia de los datos para comprimirlos y la señal descomprimida es idéntica a la original.

La compresión que realiza MPEG aprovecha la irrelevancia, desde el punto de vista de sensación sonora, de parte de la información. Esta compresión no produce la misma señal después de la decodificación, su objetivo es que el resultado sea muy parecido para el oído humano. Básicamente analiza la señal en frecuencia y elimina aquellas componentes que se ven enmascaradas por otras (no se pueden oír) y empaqueta esta señal analizada en un *bitstream* MPEG-1 audio.

Este *bitstream* tiene un formato fijo que descompone la señal en una secuencia de cuadros (*frames*) que, por norma general, son independientes unos de otros, lo que es adecuado para la difusión de audio.

Layers

Los tres *layers* definidos tienen diferentes propósitos. La complejidad del codificador y decodificador, el retardo de codificación/decodificación y la eficiencia de codificación son valores crecientes al cambiar a un *layer* mayor.

- **Layer I**: Es el menos complejo y especialmente indicado para aplicaciones en las que la complejidad del codificador es un parámetro importante.
- **Layer II**: Aumenta la complejidad del codificador y muy levemente la del decodificador. Está ideado para aplicaciones punto a multipunto.
- **Layer III**: Es todavía más complejo, elimina más redundancia y está ideado para aplicaciones que requieran una tasa binaria baja. Emplea además compresión Huffman de los datos.

MPEG-2 Audio

El estándar MPEG-2 fue terminado en 1994 y constituye el estándar internacional ISO/IEC 13818-3, publicado en 1995. MPEG-2 AAC fue terminado y publicado como el estándar internacional ISO/IEC 13818-7 en 1997.

Este estándar es la segunda fase del proceso de estandarización MPEG y consta de tres elementos bien diferenciados:

- Extensión de MPEG-1 para tasas de muestreo bajas. Incluye tasas a 16, 22.05 y 24 ksps, que originan tasas binarias por debajo de 64 kbps para cada canal. Es una extensión muy fácilmente incorporable a los reproductores MPEG-1.
- Extensión a sonido multicanal compatible con MPEG-1. Permite la utilización de hasta cinco canales con tasa binaria máxima más un sexto canal de mejora de las bajas frecuencias. Esta extensión es compatible con MPEG-1 siendo posible la reproducción de los *bitstreams* MPEG-2 BC mediante reproductores MPEG-1.
- Nuevo esquema de codificación. Este nuevo esquema se denomina *Advanced Audio Coding* (AAC) y genera *bitstreams* que no pueden ser utilizados por los reproductores MPEG-1. Mejora mucho la razón de compresión de los datos.

Layers y profiles

MPEG-1 y los dos primeros elementos de MPEG-2 contienen tres *layers*. Dada la compatibilidad de los dos primeros elementos de MPEG-2 con la primera fase, la especificación de los tres *layers* no varía.

Sin embargo, cuando hablamos de MPEG-2 AAC, que ya no es compatible con las versiones anteriores, se habla de tres perfiles (*profiles*). La diferencia radica en que los *layers* definen la estructura de la información codificada (sus tasas de muestreo, tasa de bits, etc.) y también la forma en que esta información debe ser transportada (*bitstream* formado por cuadros con una estructura fija). MPEG-2 AAC estandariza la forma que toma la información codificada pero no cómo debe ser transportada, lo que se deja a la aplicación. Por otro lado, MPEG-2 AAC, estandariza dos ejemplos típicos que pueden emplearse para el transporte de los datos:

- **ADIF** (*Audio Data Interchange Format*). Esta especificación recoge toda la información para poder decodificar los datos es una única cabecera al comienzo del *bitstream*. Está indicado para el intercambio de ficheros de audio.
- **ADTS** (*Audio Data Transport Stream*). Esta especificación divide el *bitstream* en cuadros, de forma similar a MPEG-1, que se encuentran entre dos marcas de sincronización y permiten tasas de bits variables. Está más indicado para difusión de audio en un entorno de red.

Los perfiles estandarizados para MPEG-2 AAC son tres, denominados Perfil Principal (*Main Profile*), de Complejidad Baja (*Low Complexity Profile*) y de Tasa de Muestreo Escalable (*Scalable Sampling Rate Profile*). Cada uno está indicado para unos determinados usos:

- **Perfil Principal**: Es apropiado cuando la capacidad de procesamiento y el uso de la memoria no sean parámetros críticos.
- **Perfile de Complejidad Baja**: Si alguno de los parámetros anteriores, capacidad de procesamiento o uso de memoria, son críticos; este perfil es el más adecuado.
- **Tasa de Muestreo Escalable**: Para los casos en los que se requiera un decodificador escalable.

MPEG-4 Audio

Este estándar consta de dos versiones. El estándar MPEG-4 Versión 1 se terminó en Octubre de 1998 y se publicó en 1999. La segunda versión se terminó en diciembre de 1999 y se publicó en 2000.

MPEG-4 es una fase que engloba todos los esquemas vistos para MPEG-1 y MPEG-2 define estructuras de más alto nivel por las cuales se permite el control y la combinación de elementos audio procedentes tanto de fuentes sonoras digitalizadas (p.ej. por medio de los formatos MPEG-1/2, codificación CELP, etc.) como sintetizadas (voz o sonidos musicales, tanto simples como complejos). Es por lo tanto un estándar que combina diferentes fuentes audio en un mismo *bitstream* con especial énfasis en aplicaciones multicanal, soporte multilenguaje, tasa binaria variable y recuperación de errores.

Codificación de voz en MPEG-4

La mayor novedad a los formatos de audio que incorpora MPEG-4 está asociada a codificaciones de voz con tasas binarias extra bajas. El estándar especifica el uso de codificadores paramétricos de tasa binaria extra baja para la codificación de voz, que consiguen tasas binarias comprendidas entre 2 y 8 kbps.

Los codificadores paramétricos establecen un modelo de señal que ajustan al sonido a codificar. La información transmitida son los parámetros del modelo que mejor lo ajustan a la señal. Un esquema de compresión eficiente consigue mediante este método codificar la señal de voz con calidad de conversación a tasas tan bajas como 1.5 kbps. Se espera que esta codificación sea utilizada para la implantación del servicios de telefonía generalizados sobre Internet.

Estructura de un bitstream MPEG-1, MPEG-2 y MPEG-2.5

Los *bitstreams* de audio MPEG se componen de elementos denominados cuadros (*frames*). Los cuadros, por norma general, son independientes unos de otros. Cada cuadro tiene su propia cabecera e información de audio. No existe una cabecera específica de fichero. Por lo tanto, se pueden separar cada uno de los cuadro y reproducirlos separadamente. Esto no es del todo cierto para MPEG-1 *Layer III*, donde los cuadros son habitualmente dependientes unos de otros.

Esta estructura de los *bitstreams* MPEG hace que se puedan utilizan tasas de bit variables, especificando una en cada cuadro. Esto permite que la tasa de bit dependa del contenido del cuadro, pudiendo reducir las tasas binarias cuando no impliquen una pérdida de calidad mejorando la razón de compresión de los datos.

Cabecera de cuadro MPEG

La cabecera del cuadro está constituida por los cuatro primeros bytes del mismo. Los once primeros bits valen siempre uno y constituyen la sincronización de cuadro. Este valor permite buscar por el *bitstream* las fronteras entre cuadros, ya que se trata de un byte con valor 255 seguido de un byte de valor mayor que 224. Una vez encontrado se lee el resto de la cabecera y se deben validar los datos que contiene. Cualquier valor indicado como reservado, inválido, incorrecto o no permitido debe dar como resultado una cabecera incorrecta.

La validación del primer cuadro no es suficiente. La secuencia de sincronización puede aparecer frecuentemente con ficheros binarios, por lo que se deben validar las cabeceras de varios cuadros para poder afirmar que estamos tratando con un *bitstream* MPEG audio.

Los cuadros pueden contener un CRC de 16 bits. Si este CRC está presente, siempre se sitúa justo después de la cabecera MPEG. Después del CRC se encuentra la información de audio.

CAMPO	LONGITUD (BITS)	POSICIÓN	DESCRIPCIÓN
A	11	31-21	Sincronización de cuadro. Todos los bits valen 1.
B	2	20-19	Identifica la versión de MPEG Audio 00 – MPEG Version 2.5 (1) 01 – reservado 10 – MPEG versión 2 11 – MPEG versión 1
C	2	18-17	Descripción del *layer* empleado: 00 – reservado 01 – *Layer III* 10 – *Layer II* 11 – *Layer I*
D	1	16	Bit de protección. Sólo si vale cero la cabecera va seguida de un CRC.
E	4	15-12	Especifica el *bitrate* empleado, su interpretación es función de los campos versión y *layer*. Sus posibles valores se encuentran en la tabla siguiente.
F	2	11-10	Tasa de muestreo. Sus posibles valores están recogidos en la nota (2)
G	1	9	Bit de relleno. El relleno se emplea para ajustar de forma exacta la tasa de bits escogida. Un valor 0 indica que el cuadro no lleva relleno y un valor 1 indica que se emplea un relleno de un *slot*. Un *slot* son 4 bytes para el *Layer I* y 1 byte para los *Layers II* y *III*.
H	1	8	Bit privado. Puede ser usado libremente por la aplicación. Por ejemplo para lanzar un determinado evento.
I	2	7-6	Modo de canal: 00 – Estéreo 01 – *Joint-stereo* 10 – Dual 11 – Mono
J	2	5-4	Extensión del modo. Contiene información para el modo *joint-stereo*, que simula el efecto estéreo con un único canal. (3)
K	1	3	*Copyright*. Si vale 1 indica que el sonido tiene *copyright*, 0 indica que no lo tiene.
L	1	2	Original. Si vale 0 indica que el contenido es una copia. Si vale 1 indica que el contenido es original.
M	2	1-0	Énfasis. 00 – Ninguno 01 – 50/15 ms 10 – Reservado 11 – CCIT J.17

(1) La versión 2.5 no es estándar oficial. Para reconocer esta versión se está empleando el bit 20 a cero. Esto indica una sincronización de cuadro de 11 bits. Aquellos decodificadores que no soportan esta versión esperan que el bit 20 valga 1, como si la sincronización de trama tuviese longitud 12 bits.

Inicialmente la sincronización era de 12 bits, pero se modificó para soportar esta nueva versión.

(2) Los posibles valores del campo F, que determina la tasa de muestreo (en sps) quedan reflejados en la siguiente tabla:

F	MPEG-1	MPEG-2	MPEG-2.5
00	44100	22050	11025
01	48000	24000	12000
10	32000	16000	8000
11	Reservado		

(3) El rango de frecuencias en MPEG se divide en subbandas, de la 0 a la 31. Para los *Layers I* y *II* los bits indican las bandas en las que se realiza el estéreo por intensidad según la tabla:

J	BANDAS
00	4-31
01	8-31
10	12-31
11	16-31

Para el *Layer III*, este campo indica el tipo de *joint-stereo* a realizar (intensidad o *Middle/Side*) según la tabla:

J	INTENSIDAD	M/S STEREO
00	No	No
01	Sí	No
10	No	Sí
11	Sí	Sí

Audio con bitrate variable

La cabecera de los cuadros MPEG especifica el *bitrate* empleado mediante el campo E, cuya interpretación depende de los valores de los campos versión y *layer*. La siguiente tabla muestra las diferentes combinaciones (datos en kbps):

CAMPO E	VERSIÓN 1 LAYER I	VERSIÓN 1 LAYER II	VERSIÓN 1 LAYER III	VERSIÓN 2 LAYER I	VERSIÓN 2 LAYERS II Y III
0000	Libre (1)				
0001	32	32	32	32	8
0010	64	48	40	48	16
0011	96	56	48	56	24
0100	128	64	56	64	32
0101	160	80	64	80	40
0110	192	96	80	96	48
0111	224	112	96	112	56
1000	256	128	112	128	64
1001	288	160	128	144	80
1010	320	192	160	160	96
1011	352	224	192	176	112
1100	384	256	224	192	128
1101	416	320	256	224	144
1110	448	384	320	256	160
1111	Erróneo (2)				

(1) El valor 0000 se emplea en casos en los que el *bitrate* empleado no corresponda con otro valor de la tabla. Su uso es únicamente interno, ya que otras aplicaciones no tendrían forma de determinar el *bitrate* apropiado.
(2) El valor 1111 no está permitido y debe dar lugar a una cabecera incorrecta.

La variación de este campo en la cabecera de sucesivos cuadros da lugar a un *bitstream* de tasa de bit variable (VBR: *Variable Bit Rate*). Esta posibilidad debe estar implementada en los decodificadores del *Layer III* y se recomienda su implementación para los *Layers I* y *II*.

Para el caso del *Layer II* existen combinaciones de tasas de bit y modos no permitidas. Las únicas combinaciones permitidas son las siguientes:

bitrate	MODOS PERMITIDOS
libre	Todos
32	1 canal
48	1 canal
56	1 canal
64	Todos
80	1 canal
96	Todos
112	Todos
128	Todos
160	Todos
192	Todos
224	Estéreo, *joint-stereo*, dual
256	Estéreo, *joint-stereo*, dual
320	Estéreo, *joint-stereo*, dual
384	Estéreo, *joint-stereo*, dual

Tamaño y longitud de un cuadro

El tamaño de un cuadro es el número de muestras que contiene, y es un valor constante. Para el *Layer I* es de 384 muestras y para los *Layers II* y *III* son 1152 muestras.

La longitud de un cuadro es la que tiene cuando está comprimido y se calcula en *slots*. Un *slot* son 4 bytes para el *Layer I* y 1byte para los *Layers II* y *III*. Se debe computar esta longitud para poder localizar el siguiente cuadro. Las longitudes pueden cambiar debido a variaciones en las tasas binarias.

- Para el *Layer I*: $$Longitud = 4 \cdot \left(\frac{12 \cdot bitrate}{sampleRate} + relleno \right)$$
- Para los otros dos *layers*: $$Longitud = 144 \cdot \frac{bitrate}{sampleRate} + relleno$$

MPEG Audio Tag ID3v1

El TAG es una estructura utilizada para describir un fichero de audio MPEG. Contiene información sobre el artista, título, álbum, año de publicación y género. Además contiene un espacio adicional para comentarios. Ocupa los 128 últimos bytes de un fichero de audio MPEG. Se compone de una serie de campos recogidos en esta tabla:

CAMPO	LONGITUD (BYTES)	POSICIÓN	DESCRIPCIÓN
A	3	0-2	Identifica el TAG, debe contener la cadena "TAG" si se ha incluido en el fichero y es válido.
B	30	3-32	Título
C	30	32-62	Artista
D	30	62-92	Álbum
E	4	93-96	Año
F	30	97-126	Comentario
G	1	127	Género

El género es un número que se interpreta según las correspondencias de la siguiente tabla:

0	*'Blues'*	20	*'Alternative'*	40	*'AlternRock'*	60	*'Top 40'*
1	*'Classic Rock'*	21	*'Ska'*	41	*'Bass'*	61	*'Christian Rap'*
2	*'Country'*	22	*'Death Metal'*	42	*'Soul'*	62	*'Pop/Funk'*
3	*'Dance'*	23	*'Pranks'*	43	*'Punk'*	63	*'Jungle'*
4	*'Disco'*	24	*'Soundtrack'*	44	*'Space'*	64	*'Native American'*
5	*'Funk'*	25	*'Euro-Techno'*	45	*'Meditative'*	65	*'Cabaret'*
6	*'Grunge'*	26	*'Ambient'*	46	*'Instrumental Pop'*	66	*'New Wave'*
7	*'Hip-Hop'*	27	*'Trip-Hop'*	47	*'Instrumental Rock'*	67	*'Psychadelic'*
8	*'Jazz'*	28	*'Vocal'*	48	*'Ethnic'*	68	*'Rave'*
9	*'Metal'*	29	*'Jazz+Funk'*	49	*'Gothic'*	69	*'Showtunes'*
10	*'New Age'*	30	*'Fusion'*	50	*'Darkwave'*	70	*'Trailer'*
11	*'Oldies'*	31	*'Trance'*	51	*'Techno-Industrial'*	71	*'Lo-Fi'*
12	*'Other'*	32	*'Classical'*	52	*'Electronic'*	72	*'Tribal'*
13	*'Pop'*	33	*'Instrumental'*	53	*'Pop-Folk'*	73	*'Acid Punk'*
14	*'R&B'*	34	*'Acid'*	54	*'Eurodance'*	74	*'Acid Jazz'*
15	*'Rap'*	35	*'House'*	55	*'Dream'*	75	*'Polka'*
16	*'Reggae'*	36	*'Game'*	56	*'Southern Rock'*	76	*'Retro'*
17	*'Rock'*	37	*'Sound Clip'*	57	*'Comedy'*	77	*'Musical'*
18	*'Techno'*	38	*'Gospel'*	58	*'Cult'*	78	*'Rock & Roll'*
19	*'Industrial'*	39	*'Noise'*	59	*'Gangsta'*	79	*'Hard Rock'*

WinAmp expandió esta tabla con los siguientes códigos adicionales:

80	*'Folk'*	92	*'Progressive Rock'*	104	*'Chamber Music'*	116	*'Ballad'*
81	*'Folk-Rock'*	93	*'Psychedelic Rock'*	105	*'Sonata'*	117	*'Poweer Ballad'*
82	*'National Folk'*	94	*'Symphonic Rock'*	106	*'Symphony'*	118	*'Rhytmic Soul'*
83	*'Swing'*	95	*'Slow Rock'*	107	*'Booty Brass'*	119	*'Freestyle'*
84	*'Fast Fusion'*	96	*'Big Band'*	108	*'Primus'*	120	*'Duet'*
85	*'Bebob'*	97	*'Chorus'*	109	*'Porn Groove'*	121	*'Punk Rock'*
86	*'Latin'*	98	*'Easy Listening'*	110	*'Satire'*	122	*'Drum Solo'*
87	*'Revival'*	99	*'Acoustic'*	111	*'Slow Jam'*	123	*'A Capela'*
88	*'Celtic'*	100	*'Humour'*	112	*'Club'*	124	*'Euro-House'*
89	*'Bluegrass'*	101	*'Speech'*	113	*'Tango'*	125	*'Dance Hall'*
90	*'Avantgarde'*	102	*'Chanson'*	114	*'Samba'*		
91	*'Gothic Rock'*	103	*'Opera'*	115	*'Folklore'*		

NOTA: Cualquier otro valor debe interpretarse como desconocido ('*Unknown*').

La especificación indica que se deben rellenar los campos con ceros, aunque las implementaciones no siempre lo respetan (WinAmp rellena con espacios, ASCII 32).

Para la versión ID3v1.1 se propuso un pequeño cambio. Según éste, el último byte del comentario sirve como un número de pista dentro del álbum. Si esa información se desconoce su valor debería ser cero.

Actualmente se ha estandarizado una versión 2.3 mucho más completa y versátil, pero cuyo contenido en metainformación está fuera del interés principal de este trabajo que son los formatos de audio digital. No obstante en la página Web se incluyen dos documentos que describen con precisión una versión preliminar del estándar 2.3.

9. Formato WAVE (RIFF)

El formato RIFF es un formato Windows para almacenar segmentos (*chunks*) de información multimedia, su descripción, formato, lista de reproducción, etc. El formato .WAV (*WAVeform Audio File Format*) se almacena dentro de un fichero con formato RIFF, definiendo unos segmentos concretos que puede contener. Por lo tanto es adecuado ver el formato general de un fichero RIFF y qué segmentos se definen para los ficheros .WAV.

Cabecera RIFF

El fichero RIFF lleva siempre una cabecera de 8 bytes, que identifica al fichero y especifica la longitud de los datos a partir de la cabecera (esto es, la longitud total menos 8). Esta cabecera se compone de 4 bytes con el contenido "RIFF" y los otros 4 indican la longitud. Después de la cabecera RIFF siempre hay 4 bytes que identifican el tipo de los datos que contiene; para el caso .WAV estos 4 bytes contienen "WAVE".

```
struct {
   char  id[4];
   DWORD len;
} riff_hdr;

char wave_id[4];
```

Chunks RIFF

Los datos de un fichero RIFF se componen de una secuencia de *chunks*, cada uno de los cuales consta de un campo de identificación de 4 bytes, una longitud de datos de 4 bytes y el conjunto de los datos.

```
struct {
   char  id[4];
   DWORD len;
} chunk_hdr;
```

Al procesar un fichero RIFF, se deben ignorar los segmentos desconocidos, para asegurar compatibilidad con futuras definiciones de los formatos de fichero.

Generalidades del formato WAVE

Del conjunto de segmentos que se definen para el formato WAVE existen dos que son obligatorios: el segmento de formato y el segmento de datos. Además el segmento de formato debe aparecer antes que el segmento de datos. El resto de

segmentos son opcionales, aunque como se verá más adelante el segmento FACT es obligatorio en algunos casos.

Segmento de formato (Format Chunk)

El segmento de formato identifica por "**fmt** " y se compone de dos elementos: un conjunto de campos comunes y un conjunto de campos específicos. Este último puede no aparecer, en función del formato concreto elegido.

Los campos comunes pueden representarse en la siguiente estructura:

```
struct {
      WORD  wFormatTag;
      WORD  wChannels;
      DWORD dwSamplesPerSec;
      DWORD dwAvgBytesPerSec;
      WORD  wBlockAlign;
} CommonFields;
```

wFormatTag	Indica la categoría de formato WAVE del fichero. De este valor depende el contenido del segmento de datos específicos.
wChannels	Número de canales en el segmento de datos
dwSamplesPerSec	Tasa de muestreo de cada canal
dwAvgBytesPerSec	Tasa media de bytes por segundo a la que los datos del fichero WAVE deberían ser transmitidos. Permite estimar el tamaño de buffer a emplear.
wBlockAlign	Alineación de los bloques del campo datos, en bytes. El software de reproducción necesitará procesar un múltiplo entero de este valor, por lo que permite que el buffer de reproducción esté alineado.

El software de reproducción debe permitir la existencia de campos específicos y en su caso ignorar todos aquellos campos desconocidos.

Categorías del formato WAVE

La categoría se especifica a través del campo **wFormatTag**. Los campos específicos del segmento de formato y la representación de los datos en el segmento de datos dependen de este valor. Actualmente se han definido los siguientes formatos no propietarios:

FORMATO	VALOR	DESCRIPCIÓN
WAVE_FORMAT_PCM	0x0001	Microsoft Pulse Code Modulation (PCM)
FORMAT_MULAW	0x0101	IBM μ-law format
IBM_FORMAT_ALAW	0x0102	IBM A-law format
IBM_FORMAT_ADPCM	0x0103	IBM AVC Adaptative Differential PCM format

Formato WAVE_FORMAT_PCM

Este formato define un único campo específico que indica el número de bits por muestra de cada canal:

```
struct {
      WORD wBitsPerSample;
} SpecificFields;
```

Si este campo está comprendido entre 1 y 8 bits, el valor que se almacena en una muestra es un entero sin signo. Si su valor supera los 8 bits entonces el valor almacenado en las muestras es un entero con signo.

Este campo, dividido entre ocho y redondeado al entero mayor más próximo indica el número de bytes por muestra, y permite estimar el campo **wAvgBytesPerSec** como:

$$wChannels \cdot wSamplesPerSecond \cdot BytesPerSample$$

Del mismo modo se puede calcular el valor de **wBloackAlign** de la siguiente forma:

$$wChannels \cdot BytesPerSample$$

La ordenación de los bytes de datos es la siguiente: los datos se organizan en bloques de **wBlockAlign** bytes. Estos bloques son una secuencia de las muestras para cada canal (en estéreo 0 es izquierdo y 1 es derecho), dentro de cada canal el byte menos significativo va primero.

Segmento de datos (Data Chunk)

El segmento de datos puede ser de dos tipos: tipo *data* o tipo *data-list*. El primero es una secuencia de datos sin más elementos. El segundo en una secuencia de dos tipo de segmentos: de datos o de silencio.

Un segmento tipo *data* se identifica por "**data**", mientras que un segmento tipo *data-list* se identifica por "**wavl**". Los segmentos de silencio ("**slnt**") sólo contienen un campo que es una doble palabra con el número de muestras a mantener el silencio.

NOTA: El valor de las muestras de silencio no deben ser asignadas a cero, sino al valor de la última muestra reproducida. De lo contrario, puede oírse un 'click' debido al salto que recibe el conversor D/A de salida. Es responsabilidad de la aplicación evitar este 'click' al igual que el que se puede producir al final del silencio.

Segmento FACT

El segmento FACT tiene como objetivo almacenar información importante sobre los datos del fichero. Es un segmento obligatorio en caso de que el campo de datos sea de tipo *data-list* y con cualquier tipo de compresión.

La definición básica de este segmento contiene un único campo (**dwFileSize**) pero en futuras definiciones del formato WAVE ampliará su contenido por lo que se debe usar el campo longitud de la cabecera del segmento para determinar los campos que están presentes.

Segmento Cue-Points

Este segmento ("**cue**") tiene como objetivo marcar determinadas posiciones dentro de la forma de onda que contiene el fichero. El segmento consta de un campo **dwCuePoints** indicando el número de puntos a posicionar y una secuencia de estructuras como la siguiente:

```
struct
{
      DWORD  dwName;
      DWORD  dwPosition;
      FOURCC fccChunk;
      DWORD  dwChunkStart;
      DWORD  dwBlockStart;
      DWORD  dwSampleOffset;
}
```

dwName	Identifica la posición a la que apunta. No debe coincidir con ningún otro elemento de la lista de estructuras.
dwPosition	Indica la posición de la muestra. Es el número de muestra secuencial dentro del orden de reproducción. [Ver Segmento Playlist]
fccChunk	Especifica el nombre (o identificación del segmento) que contiene al punto indicado.
dwChunkStart	Especifica la posición dentro del fichero en la que comienza el segmento que contiene al punto indicado. Es relativo al comienzo del campo de datos del segmento "**wavl**".
dwBlockStart	Especifica la posición dentro del fichero del comienzo del bloque en que se encuentra la posición apuntada, referenciada con respecto al comienzo del campo de datos del segmento "**wavl**". Su utilidad deriva en que los datos pueden estar comprimidos en bloques.
dwSampleOffset	Indica el desplazamiento del punto buscado relativo al comienzo del bloque.

Segmento Playlist

Especifica un orden de reproducción a partir de diferentes posiciones en el fichero. Se identifica por "**plst**" y contiene un campo **dwSegments**, que indica el número de segmentos que se reproducen y a continuación una lista de estructuras que describen el segmento a reproducir:

```
struct {
      DWORD dwName;
      DWORD dwLength;
      DWORD dwLoops;
}
```

dwName	Especifica el nombre de la posición en la que comienza la sección a reproducir.
dwLength	Indica el número de muestras a reproducir en esta sección.
dwLoops	Especifica el número de veces que se reproducirá esta sección.

Segmento de datos asociados

Este segmento se identifica por "**adtl**" y permite asociar información a secciones de la forma de onda del fichero. Se define como una lista de otros segmentos que son:

— Etiqueta ("**labl**")
— Nota ("**note**")
— Texto con información de longitud de datos ("**ltxt**")
— Información de fichero embebido ("**file**")

Segmentos "labl" y "note"

Su formato es similar, "**labl**" contiene una etiqueta o título asociado a una determinada posición de la forma de onda y "**note**" aporta un comentario sobre esa posición. Los dos segmentos constan de dos campos: un campo **dwName** que identifica el punto de la forma de onda considerado y un campo data que es una cadena terminada con un carácter nula que especifica el contenido de la etiqueta o la nota.

Segmento "ltxt"

Este segmento contiene información asociada a una porción de datos de una longitud determinada. Los campos de los que se compone son los siguientes:

dwName	Especifica el nombre de la posición que indica el comienzo de los datos a los que está asociado.
dwSampleLength	Indica el número de muestras de la porción de interés dentro de los datos sobre la forma de onda.
dwPurpose	Especifica el propósito del texto.
wCountry	Especifica el código del país para el texto.
wLanguage	Especifica el lenguaje empleado.
wCodePage	Indica el código de página para el texto.

Segmento "file"

Este segmento contiene información descrita en otros formatos de fichero. Su formato es el siguiente:

dwName	Especifica la posición de la forma de onda a la que se asocia el fichero.
dwMedType	Especifica el tipo de fichero que se encuentra en el campo **fileData**. Si el fichero es un tipo RIFF, este campo contiene el identificativo del tipo particular de fichero RIFF que sea.
fileData	El contenido del fichero.

10. Formato MOD

Los ficheros MOD se asemejan a los ficheros MIDI en la medida que representan sonidos musicales. Este formato se originó en plataformas Amiga, pero debido a su flexibilidad y al gran número de ficheros MOD disponibles, su uso se extendió a otras plataformas (PC, Mac, Sparc Station, etc.). Hoy en día es casi de interés meramente histórico, porque estuvo bastante extendido. Contienen dos tipos de información:

- Un banco de sonidos digitalizados
- Información para la reproducción de esos sonidos

Existen dos versiones del fichero, la inicial, en desuso, y la extendida. Las muestras digitalizadas en un fichero MOD son de 8 bits en el formato inicial y de 16 en el extendido. Carecen de cabecera y presentan codificación lineal. Puede haber hasta un total de 31 sonidos diferentes, cada uno con longitud de hasta 128k. La primera versión de ficheros MOD sólo permitían hasta 15 sonidos. No existe una tasa de muestreo estandarizada para estos ficheros.

La información sobre secuenciación de los sonidos consta de cuatro secciones, que indican:

- Qué sonido se reproduce
- Cuándo se reproduce un sonido
- Durante cuánto tiempo
- A qué tasa se reproduce

Los ficheros MOD permiten que se reproduzcan un máximo de cuatro sonidos simultáneamente, una de las limitaciones que han precipitado el abandono de su uso.

Cabecera

A continuación se incluye su formato de cabecera:

```
struct {
      char songname[20];
      Sample samples[32];
      char songlength;
      char mark;
      char songpos[128];
      char extmark[4];
}mod_hdr;
```

```
struct {
      char smpName[22];
      short smpLength;
      char fineTune;
      char volume;
      short repeatpoint;
      short repeatlen;
} Sample;
```

Los campos que describen la cabecera son:

songname	Nombre de la canción. Deje estar justificado a la izquierda y los caracteres no usados deben ser nulos.
samples	Array de 32 estructuras que determinan los parámetros de los sonidos muestreados que contiene el fichero.
songlength	Longitud de la canción. El rango está comprendido entre 1 y 128.
mark	Byte de valor 127
songpos	Es una lista de identificadores de secuencia. La canción se compone de una concatenación de pequeñas secuencias. Puede haber hasta 64 secuencias distintas, por lo que su valor se comprende entre 0 y 63.
extmark	Contiene la cadena "M.K." e identifica el fichero MOD extendido (que soporta 32 sonidos)

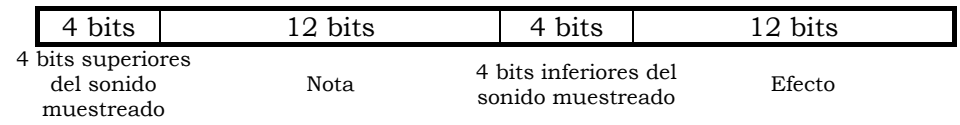
El significado de los campos de la estructura que describe un sonido muestreado es el siguiente:

smpName	Cadena ASCII que da nombre al sonido. Está justificada a la izquierda y rellenados por nulos aquellos caracteres no utilizados.
smpLength	Indica el número de palabras que contiene el sonido muestreado. Las palabras en el formato extendido son de 16 bits.
finetune	Indica el tono de la muestra. Los cuatro bits superiores deben ser nulos, los cuatro inferiores con un valor en complemento a dos.
volume	Volumen del sonido. El rango comprende de 0 a 64.
repeatpoint	Indica la muestra a partir de la cual debe repetirse el sonido.
repeatlen	Longitud de la sección del sonido que se repetirá.

Tabla de secuencias

Después de la cabecera hay una tabla de secuencias. Cada secuencia ocupa 1024 bytes. El número de secuencias está indicado por el mayor valor contenido en la lista **songpos**.

Una secuencia es una lista elementos. Cada elemento lo forman cuatro notas, una por cada uno de los cuatro canales que se pueden reproducir simultáneamente. Una nota está formada por cuatro bytes según esta organización:



Sonidos muestreados

Después de las secuencias se encuentran los sonidos muestreados. La longitud e cada uno se puede determinar por la información de la tabla de muestras contenida en la cabecera. Las muestras son de 16 bits en el formato de fichero extendido y de 8 bits en el formato inicial.

11. Formato VOC

El formato VOC es propietario de Creative y apareció con las tarjetas de sonido SoundBlaster, muy populares en plataformas PC. Es un formato de fichero muy simple pero muy poco flexible.

Todo fichero VOC debe comenzar por una cabecera de 32 bytes, tras la cabecera se encuentra una secuencia de bloques de datos, siendo el último un indicador de final de fichero (*Terminator Block*).

Los datos almacenados en un fichero VOC siguen la ordenación de bytes Intel, por la cual, en tipo de datos que consten de más de un byte, la posición más baja en memoria será la del byte menos significativo.

Cabecera

```
struct {
      char id[13];
      short offset;
      char version[2];
      char sign[2];
};
```

id Identifica el fichero como de tipo VOC. Consta de la cadena de caracteres "*Creative Voice File*" y va seguida de un carácter de valor 0x1A (Fin de fichero), que tiene como objetivo detener una posible impresión del fichero en un terminal de texto.

offset Indica la posición en la que se encuentra el primer bloque de datos. El valor estándar es 32.

version	Indica la versión según un formato "A.B" donde A está en la posición 15 y B en la 14.
sign	Es una señal que verifica la integridad de la cabecera. Se forma mediante el complemento a dos del campo versión al que se le ha sumado el valor hexadecimal 0x1234.

Bloques de datos

Tienen todos la misma estructura, salvo el bloque de terminación que únicamente se compone del campo tipo (de valor 0). El resto de bloques tiene esta forma:

```
struct {
      char type;
      char size[3];
      char info[];
};
```

type	Distingue entre los diferentes bloques de datos. Los posibles valores son los siguientes: 00 – Fin 01 – Datos de sonido 02 – Continuación de datos de sonido 03 – Silencio 04 – Marcador 05 – Cadena ASCII 06 – Repetición 07 – Fin de repetición 08 – Datos extendidos
size	Es un número sin signo de tres bytes, almacenado según la ordenación Intel, que especifica la longitud del campo de información.
info	Contiene la información del bloque. La longitud de este campo es cero para el bloque de fin de repetición

Bloques de datos de sonido

Estos bloques son los que almacenan las muestras del sonido que contiene el fichero. El formato especifica dos tipos de bloques de datos de sonido, el bloque "*Datos de sonido*" y el bloque "*Continuación de datos de sonido*". La diferencia entre ambos es que el primero incluye una pequeña cabecera de 2 bytes en la que indica la tasa de muestreo y el tipo de compresión utilizado.

- El campo tasa de muestreo se calcula como: $SR = 256 - \left(\frac{10^6}{TasaDeMuestreo} \right)$.
- El tipo de compresión puede tomar sólo cuatro valores diferentes:
  0x00 – 8 bits
  0x01 – 4 bits
  0x02 – 2.6 bits
  0x03 – 2 bits

Bloque de silencio

El bloque de silencio está formado por tres bytes que representan dos campos. El primer campo, de dos bytes, es la longitud del silencio – 1 y el segundo campo es un byte de tasa de muestreo que sirve como unidad para cuantificar la longitud del silencio.

Bloque marcador

Permite indicar posiciones concretas dentro de los datos de sonido. El *driver* software debe almacenar internamente en identificador del último marcador encontrado. Tiene un único campo, que identifica el marcador y ocupa 2 bytes.

Su finalidad es la de facilitar a las aplicaciones la edición del sonido y la composición del mismo a partir de pequeños fragmentos de sonido.

Bloque ASCII

Permite introducir una cadena ASCII en el fichero. La cadena debe finalizar con un carácter nulo. La longitud de la cadena está sólo limitada por el tamaño máximo que puede tener un bloque. La mayor cadena posible puede tener 2^{24}-2 bytes.

Bloques para repetición de fragmentos

El formato permite la repetición finita o infinita de bloques de datos. Un bloque de Repetición (tipo 6) indica el comienzo de los datos a repetir y el número de veces que estos datos deben repetirse. Este número se especifica mediante el único campo que tiene el bloque, que es de dos bytes, y toma el valor del número de repeticiones más uno. El valor 0xFFFF indica repetición infinita y el valor 0x0000 es erróneo. Los datos a repetir finalizan al encontrarse un bloque Fin de repetición (tipo 7).

Bloque de datos extendidos

Este bloque consta de cuatro bytes. El primer campo consta de dos bytes e indica una constante de tiempo que se calcula:

$$TimeCnst = \begin{cases} 65536 - \dfrac{256 \cdot 10^6}{TasaMuestreo} & \text{(mono)} \\ 65536 - \dfrac{128 \cdot 10^6}{TasaMuestreo} & \text{(stereo)} \end{cases}$$

El segundo campo, de un byte, es de empaquetamiento y el último campo, también de un byte especifica el modo (0 – mono; 1 – estéreo).

12. Otros formatos

Además de los formatos vistos hay muchos otros, todos los demás ficheros utilizan las mismas técnicas descritas en alguno de los formatos vistos anteriormente. No obstante cabe destacar la naturaleza de los siguientes formatos muy extendidos:

- **Real Audio**. No es un formato cerrado, ni define un esquema de codificación propio. En sus diferentes versiones cambia de codificación utilizada. Algunas de las versiones emplean compresión CELP y otras GSM.
- **Dolby Digital (AC-3)**. Es un formato propietario cuyo funcionamiento es muy similar al estándar MPEG. Se diferencia en proporcionar una menor tasa de codificación porque el algoritmo que emplea es más simple que MPEG-2 AAC al consideran una resolución del espectro menor.[2]
- **CD Audio**. El formato CD audio no es más que una codificación PCM de 16 bits estéreo a 44100 muestras por segundo. No tiene por tanto ningún sistema de compresión de los datos. Simplemente almacena el conjunto de muestras resultado del proceso de cuantificación sin más procesamiento.
- **OGG Vorbis**. Este formato se está difundiendo cada vez más. Se fundamenta en MPEG-2.5 y se distribuye un codificador con licencia de libre distribución.

Apéndice: Formato MIDI

El formato MIDI no se considera de audio digital, ya que no almacena muestras de un determinado sonido sino una descripción musical del mismo. Sólo puede representar sonidos musicales, además no los describe completamente, ya que indica el instrumento en el que se tocan, la forma de tocarlo, pero el sonido final que se reproduzca depende fuertemente del dispositivo reproductor MIDI.

Si bien este formato no es estrictamente audio digital, está ampliamente extendido para la representación de sonidos musicales en sistemas informáticos, por lo que le dedicamos este apéndice.

Estructura de fichero MIDI

Los ficheros MIDI tienen una estructura simple, se componen de una sucesión de segmentos. Debe haber un segmento de cabecera, seguido de un número variable de pistas (*tracks*). Una pista se asemeja conceptualmente a un pentagrama: puede describir el sonido de un determinado instrumento.

Segmento de cabecera

El segmento de cabecera de un fichero MIDI según la especificación *General MIDI 1* tiene la siguiente estructura:

```
struct {
      char id[4];
      long size;
      unsigned format;
      unsigned ntracks;
      unsigned deltatime;
}MIDIhdr;
```

id	Identifica al fichero como de formato MIDI. Contiene la cadena de caracteres "MThd".
size	Indica la longitud de la cabecera del fichero MIDI, sin contar los campos **id** y **size**. En esta especificación la

[2] La resolución del espectro es un parámetro fundamental de la compresión perceptiva, es la unidad elemental que se emplea para analizar la señal en frecuencia. Una resolución mayor es más costosa computacionalmente pero define más exactamente la señal y permite una compresión mayor.

	cabecera siempre tiene 6 bytes.
format	Hay tres posibles formatos: 0 – Pista única 1 – Multipista, síncrono 2 – Multipista, asíncrono
deltatime	Indica el número de intervalos de tiempo contenidos en un cuarto de nota. Especifica así el tempo de la reproducción.

Formatos

- Pista única: Indica que el fichero se compone de cabecera y un único segmento de pista.
- Multipista, síncrono: Especifica un fichero con múltiples pistas, todas ellas refrerenciadas a tiempo cero. La reproducción de las pistas es por tanto simultánea.
- Multipista, asíncrono: El fichero contiene múltiples pistas que se reproducen de forma secuencial, una detrás de otra.

Segmentos de pista

Cada segmento de pista consta de una cabecera, similar a la del fichero, seguida de una serie de eventos MIDI. De entre los eventos MIDI se encuentran los comandos, que son los mismos datos enviados o recibidos por los puertos MIDI de los dispositivos.

```
struct {
      char id[4];
      long size;
}MIDItrackhdr;
```

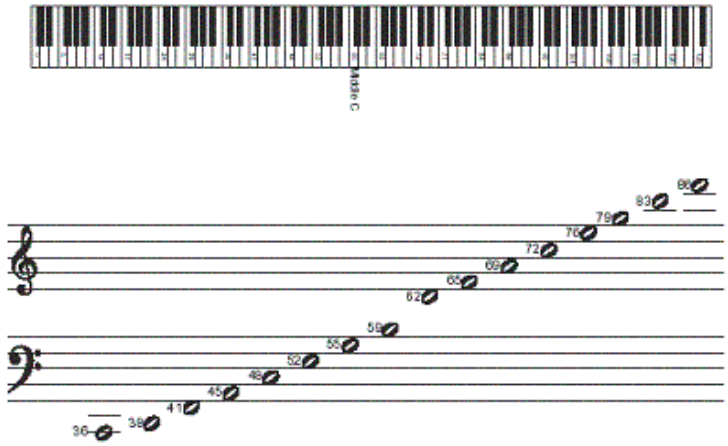
id	Es una cadena de cuatro caracteres que especifica el tipo de segmento como una pista MIDI. Su contenido es "**MTrk**".
size	Indica el tamaño de la pista sin contar la cabecera, esto es, el tamaño del conjunto de comandos que contiene.

Los programas que interpreten ficheros MIDI deben considerar la posible existencia de otros segmentos aparte de los de cabecera o pista y deben ignorarlos. Esto permite una mayor libertad a la hora que introducir nuevos elementos a la especificación MIDI.

Notas musicales en MIDI

Las notas musicales se representan en MIDI mediante un número entero positivo comprendido entre 0 y 127 (requiere de 7 bits). Con este rango cubre 11 octavas. Las correspondencias son las siguientes:

OCTAVA	C DO	C#	D RE	D#	E MI	F FA	F#	G SOL	G#	A LA	A#	B SI
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				



Valores de longitud variable

En muchos de los comandos MIDI se empleará una representación de números que se denomina de longitud variable. La intención es representar un rango amplio de valores sin obligar a que valores pequeños ocupen más bytes los que el valor necesita para ser almacenado.

La forma de representación es la siguiente. El número se formará como una secuencia de bytes. Cada byte aporta sólo siete bits (los menos significativos) al número representado. El bit de mayor peso debe valer 1 si quedan más bytes para representar el número y 0 para el último byte. Estos son algunos ejemplos de valores hexadecimales y su representación en longitud variable MIDI:

VALOR HEXADECIMAL	REPRESENTACIÓN MIDI
0x0000000	00
0x0000040	40
0x000007F	7F
0x0000080	81 00
0x0002000	C0 00
0x0003FFF	FF 7F
0x0004000	81 80 00
0x0100000	C0 80 00
0x01FFFFF	FF FF 7F
0x0200000	81 80 80 00
0x8000000	C0 80 80 00
0xFFFFFFF	FF FF FF 7F

Eventos

Los eventos contenidos en los segmentos de pista de un fichero MIDI constan de dos elementos: un campo *delta-time* y un campo evento. El campo *delta-time* es de longitud variable y se expresa una determinada fracción del compás, tal como se indica en la cabecera. El campo evento puede ser de tres tipos:

- Evento MIDI: Es un mensaje de un canal MIDI. El primer evento de una pista debe ser un indicador de estado. El estado se almacena, de forma que no es necesario volver a indicarlo si el mensaje de canal MIDI anterior tenía el mismo estado.
- Evento Exclusivo del Sistema MIDI: Especifica un mensaje exclusivo del sistema MIDI.
- Eventos "meta": Especifica información que no es MIDI, útil para los secuenciadores.

Eventos MIDI

De los eventos MIDI se distinguen los comandos de voz de los comandos de modo.

— COMANDOS DE VOZ: Constan de un byte de estado seguido de unos datos dependientes del byte de estado. El sistema MIDI organiza los diferentes instrumentos en canales. Hay 16 posibles canales que se especifican con los 4 bits más bajos del byte de estado. Los posibles mensajes son:

BYTE DE ESTADO	SIGNIFICADO	DATOS
0x8-	Liberación de nota	1 byte de tono 1 byte de velocidad
0x9-	Pulsación de nota	1 byte de tono 1 byte de velocidad
0xA-	Presión de tecla	1 byte de tono 1 byte de presión (después de pulsar)
0xB-	Parámetro	1 byte de número de parámetro 1 byte de valor
0xC-	Programa	1 byte de programa seleccionado
0xD-	Presión del canal	1 byte de presión (después de pulsar)
0xE-	*Pitch wheel*	2 bytes que proporcionan 14 bits, los 7 de menor peso primero.

El sistema MIDI almacena el último byte de estado, de forma que si se quiere transmitir un tipo de mensaje, por el mismo canal, no es necesario volver a mandar el byte de estado.

La velocidad es para aquellos dispositivos que son sensibles a la pulsación de la tecla a diferentes velocidades. Su valor está representado en escala logarítmica. Los dispositivos que no tienen esta funcionalidad se suponen que pulsan la tecla con una velocidad igual a 64.

— COMANDOS DE MODO: Son comandos que constan sólo de un byte, es byte de estado pero no especifican ningún canal concreto. Este grupo es poco numeroso y el comando más utilizado tiene como función silenciar la totalidad de los canales.

Mensajes Exclusivos del Sistema MIDI

Los eventos exclusivos pueden ser únicos, en paquetes (una secuencia) o una secuencia de escape. Existen fundamentalmente dos forma de mandar este tipo de mensajes:

0xF0 + longitud + mensaje

El campo longitud especifica el número de bytes del mensaje (no incluye cabecera ni el propio campo longitud). Es un valor expresado en formato longitud variable. El mensaje completo puede enviar en un solo bloque o en varios. El mensaje completo debe finalizar con el carácter 0xF7 (que no se considera parte del mensaje). Si una mensaje contenido en una secuencia 0xF0 no finaliza con el carácter 0xF7, consta de más secciones. El resto del mensaje completo se envía con secuencias similares pero que comienzan con el carácter 0xF7.

Existe otra forma de enviar datos a modo de secuencia de escape. Este es un método para transmitir datos que de otra forma no serían válidos. Se trata de una secuencia como las vistas, que comienza por 0xF7, pero no es continuación de una secuencia 0xF0:

0xF7 + longitud + mensaje

Eventos META

Los eventos meta son muchos, no son información MIDI sino otros tipos de información que asociamos al fichero MIDI. Los programas de aplicación no necesitan saber interpretar todos los eventos meta. A continuación se especifican algunos eventos meta:

EVENTO META	DESCRIPCIÓN
FF 00 02 secnum	Número de secuencia. Identifica una secuencia, para poder componer canciones en el formato 2 mediante la sucesión de secuencias que se identifican por este número. Es opcional y de existir debe situarse al principio de una pista. Antes de cualquier *delta-time* no nulo y de cualquier evento MIDI transmisible. Los ficheros de tipo 0 o 1, que sólo constan de una secuencia, deben tener este campo en la primera (o única) pista.

FF 01 long texto	Texto. Permite introducir un texto en el fichero MIDI. Normalmente se suele incluir al comienzo de una pista para describir la forma en la que se debe representar. Pueden también emplearse en otros puntos del fichero, pudiendo servir como letras o bien como explicaciones de marcadores dentro de la pista. Se debe componen con caracteres ASCII imprimibles y aquellos programas que no los soporten deben ignorarlos. El campo **long** es de longitud variable.
FF 02 long texto	Nota de *copyright*. Incluye una nota de *copyright*. La epecificación indica que debe contener los caracteres "(C)", el año del *copyright* y el propietario del mismo. Si existen diversas piezas musicales, los *copyright* deben agruparse en un único texto por fichero.
FF 03 long texto	Nombre de secuencia/pista. Indica para los ficheros de formato 0 o la primera pista de los de formato 1 el nombre de la secuencia. En otro caso lo que indica es el nombre de la pista.
FF 04 long texto	Nombre de instrumento. Describe el tipo de instrumentación a usar en la pista. Puede ir utilizado con el meta-evento Prefijo de canal MIDI para que indique el canal al que aplica la descripción.
FF 05 long texto	Letra Es una letra para ser cantada. De forma general, cada sílaba es un evento Letra distinto, que comienzo en el instante del evento.
FF 06 long texto	Marca Se utiliza normalmente en la pista del formato 0 o bien en la primera pista del formato 1. Nombra un determinado punto de la secuencia (p.ej. "Verso primero").
FF 07 long texto	Apunte Indica algo que sucede en una película o pantalla de vídeo en el momento en el que la música se encuentra en ese instante de la reproducción.
FF 20 01 cc	Prefijo de canal MIDI. El canal MIDI que indica (de 0 a 15) se toma como canal actual, de forma que los eventos siguientes están asociados a este canal (tanto los de sistema como los meta-eventos).
FF 2F 00	Fin de pista. Este evento es obligatorio. Se incluye para especificar un punto exacto de finalización, muy importante sobre todo si las pistas se repiten o se concatenan.
FF 51 03 tempo	Tempo. Indica un cambio en el tempo en microsegundos por cuarto de nota MIDI.
FF 58 04 nn	Tiempo.
FF 7F long texto	Evento meta de secuenciador. Se emplea para definir mensajes específicos de un determinado secuenciador. El primer byte del mensaje es el ID del fabricante. Pero si el primer byte es cero, el campo de identificación es de tres bytes. Los formatos de estos mensajes específicos deben ser publicados por cada fabricante para permitir a otros secuenciadores la interpretación de esa información.

Utilidades software

Para trabajar con estos formatos se emplean multitud de utilidades software, para el manejo de estos formatos de audio, su reproducción, su edición y modificación. Este apartado se centra en utilidades de edición y conversión.

En entornos Windows, destacan aplicaciones como *CoolEdit*, *AudioEdit* o *AudioConvert*.

- **CoolEdit** (ce2kmain.exe). Este programa, el más extendido de los tres, es el que más formatos soporta: 14 formatos en total, destacando la inclusión de formatos WAVE con compansión (Ley-A o Ley-Mu). Además de filtros en frecuencia, ecualizaciones, conversiones entre formatos y codificaciones, permite también un análisis espectral de la señal audio. La versión contenida en la página WEB es para plataformas Windows 2000/XP. Su licencia es shareware y sólo permite ejecutar el programa con un conjunto de capacidades determinado.
- **AudioConvert**. Permite la grabación de audio con formatos de salida: WAV, G.723, G.726, VOX, MP2, MP3, WMA y OGG Vorbis. Permite tasas de muestreo comprendidas entre 8 y 48 ksps, para canalización mono y estéreo. Además permite modificar señales en formato WAV, mediante filtrados en frecuencia, ecualización, cambios en la tasa de muestreo, adición de ruidos y otros. Permite la conversión entre cualquier par de formatos soportados. Existe una versión de evaluación por 14 días en la página web.
- **AudioEdit**. Permite formatos WAV PCM, VOX, OGG, MPEG (varios), WMA y formatos sin cabecera. Permite el cambio de tasa de muestreo, la inclusión de efectos como el eco, filtros en frecuencia y amplitud, ecualización, etc. Se encuentra una versión de evaluación por 15 días en la página web.

Para UNIX se emplean utilidades similares. Una de las más importantes es SOX, que es una biblioteca de código para la conversión entre formatos. SOX, que no sólo es utilizable en plataformas UNIX, contiene un conjunto de módulos '.c' con código para la interpretación de muchos formatos de audio.

El objetivo de SOX es la conversión entre muestras de uno y otro formato y por esta razón no contempla los formatos que no representen una secuencia de muestras. Entre estos formatos se encuentran los de compresión perceptiva, entre ellos los formatos MPEG.

En la página WEB se han incluido dos ficheros relacionados con SOX. El primero (sox10dos.zip) contiene un ejecutable para entornos MS-DOS del programa. El segundo fichero (soxgamma.tar.gz) contiene los ficheros fuente del programa de la última versión disponible de SOX.

Además de estas utilidades se ha incluido en la página el fichero adpcm.zip que contiene un conjunto de módulos en lenguaje C para codificar y decodificar los formatos G.711, G.721 y G.723 del CCITT.

Conclusiones

Los formatos de audio digital permiten almacenar y describir sonido codificado digitalmente. Inicialmente surgieron multitud de formatos, en principio sin estructurar y más adelante ficheros estructurados cada vez más versátiles.

La exigencia de mayor calidad en el sonido almacenado aumentó considerablemente el tamaño de los ficheros mostrando la necesidad de

compresión. Las primeras estrategias se basaban en la redundancia de los datos, una de estas estrategias es ADPCM (codificación diferencial). Las técnicas actuales, con mayores razones de compresión se fundamental en la irrelevancia de muchos de los datos y modifican la señal en función de la respuesta del oído humano al sonido.

Los formatos más relevantes en la actualidad están estructurados y permiten una gran variedad de codificaciones diferentes. La tendencia actual indica un desarrollo mayor aún de formatos con compresión perceptiva, relegando a los formatos basados en PCM para la utilización en los procesos de síntesis y manipulación de sonido previos a una conversión de formato para el almacenamiento o la difusión.

Se espera que el sonido digital vaya relegando los sistemas de difusión de audio analógicos actuales. Un ejemplo es la radio digital por modulación de amplitud, sistema para el cual MPEG-4 con una configuración concreta ha sido validado y es una de las posibles opciones para la implantación de este servicio.

Referencias

Las siguientes direcciones de Internet contienen información relacionada con los formatos de audio digital. Algunas amplían los puntos que en este trabajo se han tratado más por encima.

- "*The Programmer's File Format Collection*" (http://www.wotsit.org). Contiene una colección muy numerosa de descripciones de formatos de fichero. De interés para nosotros es la sección para formatos de sonido.
- "*MIDI Technical/Programming Docs*" (http://www.borg.com/~jglatt/tech/miditech.htm). Diversa información sobre el formato MIDI.
- "*MPEG Audio Public Documents* " (http://www.tnt.uni-hannover.de/project/mpeg/audio/public). Contiene los documentos que conforman los estándares MPEG Audio.
- "*Fraunhofer IIS-A - AMM - Technology Information*" (http://www.iis.fhg.de/amm/techinf/index.html). Ofrece una descripción general del estándar MPEG-2.5

A continuación explico el contenido de los documentos de información complementaria que he incluido en la página WEB. Parte de la información de este trabajo puede encontrarse en estos documentos, pero su función principal es la de aclarar aquellos aspectos que no haya dejado claro y permitir profundizar más sobre alguno de los temas relacionados con los formatos de audio digital.

- "*Audio Interchange File Format AIFF-C*" (aiff-c.9.26.91.pdf). Como indica el subtítulo es una revisión del estándar Audio IFF para incorporar capacidades de compresión de los datos. El documento que se recoge en un borrador del estándar final.

- Estándar ISO-11172 (iso11172.zip). Contiene cuatro documentos Word que tienen la finalidad de describir con claridad el *bitstream* MPEG-1 y cómo se realiza la decodificación del mismo.

- "*MP3 and AAC explained*" (3-1.pdf). Es un documento del *Fraunhofer Institute for Integrated Circuits* que explica muy claramente, y de forma conceptual las diferencias entre los diferentes formatos de audio MPEG. También explica el algoritmo utilizado, las mejoras introducidas entre versiones y realiza un estudio de la calidad de audio obtenida.

- "*Report on the MPEG-2 AAC Stereo Verification Tests*" (w2006.pdf). Es un informe ISO sobre unas pruebas realizadas para comparar la nueva codificación AAC con las anteriores versiones de MPEG. Especialmente interesante es el apartado de conclusiones donde se constatan los mejores resultados obtenidos por AAC para tasas de bits iguales o inferiores a las otras versiones.

- "*Revised Report on Complexity of MPEG-2 AAC Tools*" (w2957.pdf). Realiza un análisis de la complejidad asociada a las diferentes utilidades que conforman el algoritmo de codificación avanzada de audio (AAC). El análisis se realiza en función del número de instrucciones a ejecutar, el número de posiciones de lectura/escritura necesarias y el número de posiciones de sólo lectura necesarias.

- Estándar ISO/IEC 14496-3 (w2803.pdf, w2803_i.pdf, w2804_n.pdf). Constituye la especificación del estándar MPEG-4 Audio. El primero de los ficheros es meramente la portada. El segundo es el contenido informativo y el tercero el contenido normativo. Los estándares MPEG tratan de minimizar los elementos normativos.

- "*ID3 tag version 2.4.0*" (id3v2.4.0-structure.txt, id3v2.4.0-frames.txt). Estos ficheros son la especificación de la estructura TAG versión 2.4.0. Esta estructura contiene datos relativos a un fichero audio. En la actualidad se usan las versiones 1 y 1.1 en ficheros MPEG-1/2 *Layer III*.

- Otros formatos para el fichero WAVE (wavecomp.zip). Este fichero describe las particularidades de otros formatos contenidos en ficheros WAVE aparte del WAVE_FORMAT_PCM descrito en este documento.

- Formatos de ficheros musicales (musfmt10.zip). Contiene la descripción de un conjunto de formatos que pueden especificar sonidos musicales. Destacan los formatos MIDI y MOD. Para este último describe diferentes efectos de sonido que se pueden incluir y que no se han precisado en este documento. Además contiene muchos otros, sobre todo asociados a programas secuenciadores, que sólo son usados por ese programa concreto.

- Comandos MIDI. (fmidi312.zip). Este fichero comprimido contiene dos documentos HTML. Uno proporciona una descripción de entradas y salidas del sistema MIDI y el segundo aporta una descripción muy detallada de algunos de los comandos MIDI no tratados en este trabajo.

Bibliografía

Estos son algunas de las referencias bibliográficas recomendadas para profundizar en el tema.

— Para el formato MIDI:

- FRANCIS RUMSEY. "*MIDI Systems and Control*" 1990 Focal Press

- BERND ENDERS y WOLFGANG KLEMME. "*MIDI and Sound Book for the Atari ST*" 1989 M&T Publishing, Inc.

- Especificaciones de fichero MIDI y del estándar MIDI General. Pueden obtenerse enviando un correo a la dirección

LISTSERV@AUVM.AMERICAN.EDU con un mensaje que contenga: GET MIDISPEC PACKAGE

— Para las codificaciones MPEG-1

- ISO/IEC 11172-3. "*Coding of Moving pictures and associated audio for digital storage media at up to 1.5 Mbit/s - Audio Part*". International Standard, 1993. Se encuentra en la documentación complementaria.

- BRANDENBURG, K.-H.; STOLL, G: "*ISO-MPEG-1 Audio: A Generic Standard for Coding of High Quality Digital Audio*". Journal of the Audio Engineering Society, Oct. 1994, Vol. 42, No. 10, pp. 780 - 792.

- DAVIS PAN: "*A Tutorial on MPEG/Audio Compression*". IEEE Multimedia Vol. 2, No. 7, 1995, pp. 60-74.

- Capítulo 4 en HASKELL/PURI/NETRAVALI: "*Digital Video: An Introduction to MPEG-2*". Chapman & Hall, Nueva York, 1997.

- PETER NOLL: "*MPEG Digital Audio Coding*". IEEE Signal Processing Magazine, Sept. 1997, pp.59-81.

- KARLHEINZ BRANDENBURG: "*MP3 and AAC explained*". Proc. of the AES 17th International Conference on High Quality Audio Coding, Florencia, Italia, 1999. Se encuentra en la documentación complementaria.

— Para las codificaciones MPEG-2:

- ISO/IEC 13818-3. "*Information Technology: Generic coding of Moving pictures and associated audio - Audio Part*". International Standard, 1994.

- ISO/IEC 13818-7. "*MPEG-2 advanced audio coding, AAC*". International Standard, 1997.

- M. BOSI, K. BRANDENBURG, S. QUACKENBUSH, L. FIELDER, K. AKAGIRI, H. FUCHS, M. DIETZ, J. HERRE, G. DAVIDSON, Y. OIKAWA: "*ISO/IEC MPEG-2 Advanced Audio Coding*", Journal of the AES, Vol. 45, No. 10, Octubre 1997, pp. 789-814

- KARLHEINZ BRANDENBURG: "*MP3 and AAC explained*". Proc. of the AES 17th International Conference on High Quality Audio Coding, Florencia, Italia, 1999. Se encuentra en la documentación complementaria.

Para más información sobre el formato Audio IFF, referirse a Apple Corporation (www.apple.com). Para el formato WAVE, más información en Microsoft Corporation (www.microsoft.com). Para más información sobre el formato IFF referirse a Electronic Arts.