卒業論文
MP3 エンコーダのハフマン符号器を用いたエンコード処理の高速化

東海大学情報理工学部ソフトウェア開発工学科

6ADF1118　岡部裕之

２０１０年1月３１日

東海大学　情報理工学部
ソフトウェア開発工学科
学生番号：6ADF1118
氏名　　：岡部　裕之

題：MP3 エンコーダのハフマン符号器を用いたエンコード処理の高速化

１、本研究の意味

本研究は、1989 年に開発された圧縮音声情報規格「MPEG Audio-Layer 3（以下 MP3.）」のエンコード処理の高速化をはかることを目的とした研究である。この研究の最終目標は、リアルタイムで音声情報をエンコードすることである。この研究が完成すれば、高音質である情報をより遠くに、より早く届けることが出来る他、インターネットの音楽配信コンテンツでの活躍など、様々な分野において活躍することが予想される。

２、MP3 とは

1970 年代初頭、ドイツのエルランゲン-ニュルンベルク大学のディター・ザイター（Dieter Seiter）教授が、電話回線を通して音楽を圧縮転送するという課題を研究したことが MP3 の起源となる。後、MP1（MPEG Audio-Layer 1）規格と MP2（MPEG Audio-Layer 2）規格を通して、それらにハフマン符号化アルゴリズムを取り入れた規格が 1989 年に誕生した。まだ、明確な名前や規格として確立していなかった。そして、パーソナルコンピュータが一般に普及し始めた 1995 年に規格に MP3 という名前がついた。衛星放送の音声送信フォーマットにも選ばれた。その後、インターネットが普及することにより電子情報をより遠くに届けることが容易になったことで、CD 音源を MP3 に圧縮し他人に配信する行為が流行る。また、MP3 は CD 音源の 10 分の 1 のファイルサイズで、CD 音源と同じ程度の音質を維持できるため、小型のポータブルオーディオ再生機が出来る。現代では、アップルコンピュータの「i-Pod」やソニーの「ウォークマン」を代表する所謂ポータブルオーディオプレイヤーが人気となっている。それらが再生することが出来るオーディオ規格の中心はもちろん MP3 である。また、携帯電話にオーディオ再生機能を付けたものまである。このように、MP3 は一般大衆に深く浸透している。

なぜ MP3 は普及したのだろうか。MP3 が開発された時代は、インターネットの回線は電話回線であり、非常に転送速度が遅く、インターネットの利用者数も少なかった。Windows95 が出てきた頃、Microsoft 社の「Internet Explorer」が OS に標準で付属されたのをきっかけに、インターネット利用者数が増え、個人でホームページを持つ人が増えだ。そして、そのホームページで自分の作曲した音楽を配布すると言う行為が可能となった。この頃は、MP3 よりも MIDI という規格で配布する形が主流だった。MIDI とは、楽譜とそれを演奏する演奏者を電子化したようなもので、これを再生する特別な再生装置で音を鳴らすと言う方式だった。ファイルサイズは 1 メガバイトもなく、通信速度が遅い電話回線でもそんなに時間をかけずにダウンロードできた。しかし、インターネットの通信速度が上がっていくにつれて、より大きなファイルサイズのデータをダウンロードすることが可能となり、ISDN 回線の普及により、

国際電話料金をかけずに海外のインターネットサイトに接続できるようになったことから、利用方法の幅が増えた。そして、自分の気に入った曲の CD を MP3 に変換し、自分のインターネットサイトで公開し、その曲を広めようという行為が出来るようになった。そして、小型で持ち運べる MP3 再生装置が開発され、従来のテープ、CD、MD を再生する所謂「ウォークマン」を置き換えていった。代表的なものにアップルコンピュータの「i-Pod」がある。この「i-Pod」が、さらに MP3 を広めることになる。

i-Pod の登場により、ポータブルオーディオが変わった。今までは、買ってきた（もしくは CD レンタルショップ等で借りてきた）CD から MD やテープに録音し、それを専用のポータブル再生機器にて、音楽をどこでも楽しむと言うスタイルだった。この場合、コンポーネントステレオや何かしらの録音機を通じて、録音媒体に録音することになる。MD の場合、CD の PCM から AAC という圧縮規格に変換されて MD に記録される。この時、MP3 と同じで可視聴音域外の音をカットする方式を取っているので、音質はすこし劣化する。MD から MD へ録音することを繰り返すと、ノイズが激しくなり、データは劣化する。テープはデジタルではなくアナログである。こちらも、何回もダビング（テープからテープへの録音、もしくは録画）を繰り返すと、データが劣化してノイズが乗るようになる。この二つの方式では、分計算で録音できる容量が決まっている。MD の場合、1 倍速（標準）モードでは、多くて 80 分、4 倍速でその 4 倍の時間、8 倍速では 1 倍速の 8 倍の時間。ただし、音質は倍速にすればするほど、劣化する。テープも同じである。MD の場合は、圧縮レートを上げることにより、一曲のデータ容量を少なくしている。テープでは、倍速録音をする時、テープの回転をその分遅くすることにより、録音時間を長くしている。

一方、i-Pod はパソコンに USB 端子で繋ぎ、専用のソフトウェアでパソコンの HDD 上にある音楽ファイルを i-Pod 本体に転送するだけで、どこでも音楽を楽しむことが出来る。扱えるファイルは MP3 だけではなく、WAVE ファイルや AAC も扱うことが出来る。MP3 は CD と同等程度の音質で 10 分の 1 程度のファイルサイズまで圧縮することが出来る。MD とは違い、専用の録音機が必要でなく、パソコンを使って簡単に録音ができ、録音している曲が最後まで終わるのを局の時間と同じ時間待つ必要もない。CD アルバムを MP3 に圧縮するのに長くても 10 分で済む。さらに、i-Pod 本体のメモリ容量も大きかった。今のように 160 ギガバイトなんて大きな容量ではないものの、MP3 を 100 曲入れるには十分な容量だった。CD を沢山持っている人は、その分の MD やテープを用意する必要がなくなったのである。これ一つで全部収まってしまうのである。

しかし、MP3 ファイルを取り扱うことが出来ると言うことは様々な問題を起こした。特に、一番大きな問題は、著作権の侵害である。前述したとおり、当時はインターネット上に MP3 ファイルを公開するという行為が行われていた。しかし、この行為は音楽のアーティストへの著作権を侵害する行為として問題になっていた。i-Pod が出る前までは、パソコンを使い音楽を聴くスタイルをとる人たちが MP3 を多く利用していた。音楽を配信、配布する人たちも利用していた。更に、「WinMX」「Winny」などの P2P（Pear To Pear）通信

ソフト（所謂、交換ソフト、共有ソフト）を利用して、不正に人の著作物を交換、共有する行為が蔓延していた。音楽も例外ではなかった。i-Pod が世界中に広まり始めた頃、このソフトを利用して音楽ファイルをゲットしようとする人々が増えた。無料で好きなアーティストの曲が入手できるソフトがある、という噂が広まった結果、i-Pod も売れていった。MP3 はその波に乗り、世の中に浸透していった。デジタルデータであり、パソコンで簡単に扱えると言うことは、コピーしても劣化しないデータである。何度コピーしても、劣化しない。MD やテープから比べればかなり便利になった。

MP3 は曲の著作権だけでなく、エンコーダそのものにも特許権がある。MP3 は圧縮方式にハフマン符号を利用している。このハフマン符号化する部分は、非常に複雑であり、過去の多くの MP3 エンコーダは ISO 仕様書規格通りに作られた「dist10」という MP3 エンコーダのハフマン符号化アルゴリズムを元に作っていた。しかし、このハフマン符号化アルゴリズムの特許を取った会社が現れたためインターネットでフリーソフトとして配布する形を取っていた多くの MP3 エンコーダは特許の使用料金を払うことが出来ない、または裁判所からの警告や会社からの警告によって、撤退することを余儀なくされてしまった。このハフマン符号化アルゴリズムの特許を回避して MP3 にエンコードするためには、特許方式とは違うやりかたでエンコードするほかない。この違うやり方で MP3 にエンコードすることを成功させたのが「LAME（LAME Ain't an MP3 Encoder = “LAME は MP3 エンコーダではない”）」である。実は、MP3 エンコーダには特に決められたエンコード方法は仕様で定められていないのである。つまり最初からエンコードする方法は、自由だったのだ。MP3 デコーダで読むことができるファイルにしてしまえば、それはもう MP3 なのである。特に LAME は“力技”ともいえるやり方で、特許を回避しているだけでなく、他のエンコーダよりもいくらか音質が良いという評判がある。さらに、エンコード時に音質を調整できるオプションや、ショートブロックのみ、ロングブロックのみでエンコードできるなど、様々なオプションが用意されている。現在出回っている殆どの MP3 はこの LAME でエンコードされている。また、LAME エンコーダのエンコード速度を早くした「午後のコーダ」というエンコーダもある。「午後のコーダ」は一時期、日本のインターネット上で爆発的な人気を誇っていた。GUI 環境で LAME を使ったエンコードが出来たことが人気になった理由の一つであると考えられる。DLL（Dinamic Link Library）版もあり、他のソフトウェアで MP3 をエンコードする、又はデコードする場合、この DLL 版を組み込んでいたソフトウェアが多かった。午後のコーダでエンコードした MP3 ファイルをバイナリエディタで開くと、いたるところに「GOGO」という文字が埋め込まれているのが特徴である。この二つのエンコーダは、コンパイル済みのものを配布せず、コンパイルする前のソースコードとコンパイラを配布している。こうすることで、研究目的のオープンソースソフトウェアという建前で、特許を回避している。

MP3 には特徴がある。前述したハフマン符号化による圧縮もその一つと言える。MP3 は人間には聞き取ることの出来ない高い音域を削ることにより、更にファイルサイズを小さ

くしている。しかし、高音域を削ることは、音質に対して悪影響を及ぼす。たとえば、音が妙に狭く聞こえるとか、迫力がなくなるとか、爽快感がなくなるとか。それをある程度防止する方法として、心理聴覚音響モデルという人間の錯覚を利用する方法がある。

３、MP3 の仕様

前述したとおり、MP3 はエンコードの仕方を仕様で決めていない。デコードできれば良いとされている。ISO 仕様書にはエンコード時に、デコードする時に必要な情報の項目が書かれている。ISO 仕様書通りに作られた Dist10 でエンコードした MP3 ファイルは、ISO 仕様書に書かれている情報の項目の順番にビットを出力しているため、仕様書をみれば人でも解読できる。なので、私は仕様書と照らし合わせながら Dist10 でエンコードした MP3 ファイルを手で解読していった。時間はかかったものの、何とか基本の構造は理解することが出来た。

さて、あなたが Dist10 でエンコードしたＭＰ３ファイルを期待に胸を躍らせながら、バイナリエディタを使い開いた時、そこには意味不明な数字とアルファベットがずらりと並んでいるだろう。そしてあなたは驚愕し、恐れ、逃げたくなるだろう。しかし、その文字列にはそれぞれ意味があることを忘れてはいけない。森を見て木を見ず。その木にはいろいろな情報が書かれていて、あなたを MP3 エンコードという目的に導いてくれる。そして、あなたは右手にペンと消しゴムを、左手には「ISO/IEC 11172-3」と書かれた一冊の地図とノートを持ち、果敢にその文字列の森に挑むのだろう（笑）。正直、この作業には相当な気力と根気が必要である。しかし、もしあなたが仕様書を見ただけで理解が出来、C 言語なりのプログラムが書けるのならば、その仕様書にそってプログラムを作り、そのプログラムを使い解読するほうが労力は大分減るだろうとコメントしておく。ちなみに、文字列の森と述べたものをビットストリームと呼ぶ。本当は文字列ではなく、バイナリコードであったりする。

さて、まずは MP3 ファイルの基本構造から説明する。MP3 は「フレーム」という大きなくくりの中に「グラニュール」という中くらいのくくり、そして「チャネル」という小さなくくりによって構成されている（図１）。

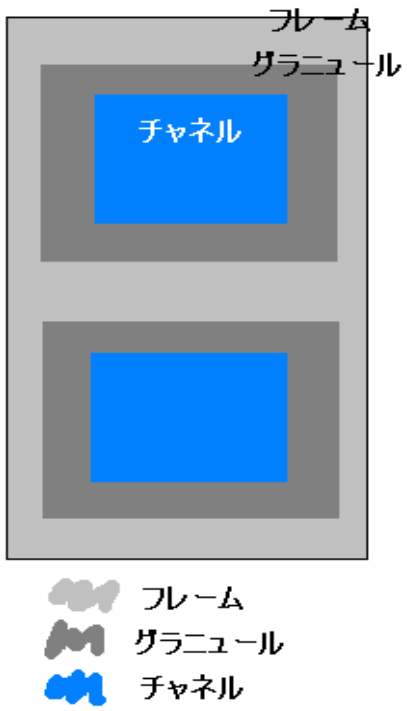


図１、MP3 の１フレームの構成図

チャネルはモノラルの場合は１つ、それ以外は２つある。グラニュールは必ず２つある。このグラニュールとチャネルの情報を持った部分がフレームとなる。このフレームをたくさん並べたものが MP3 である。1 フレーム含

まれるバイト数をＮとすると次の計算でＮを求めることが出来る。

$$N = 144 \times \frac{bitrate}{sampling\_frequency}$$

今、ここに bitrate が 128kb/s で sampling_frequency が 44,1kHz の固定レートのＭＰ３データがあるとする。上の式を使い計算するとＮ＝418（四捨五入）．つまり、1 フレーム中に 418 バイトの情報があると言うことである。この 418 バイトの情報の中に元の波形にもどすための情報が含まれている。

フレームの一番最初にはフレームの初めを標している同期語「Syncword」がある。この同期語を使い、フレームの最初を探す。同期語は 12bit の”1”で構成されている。つまり、１が 12 個並んでいるだけである。バイナリエディタであれば、16 進数なので“FFF”となる。この同期語のあとに、デコードに必要な情報が記述されている。

同期語を含む最初の４バイトのデータをヘッダと言う。このヘッダには連続して「同期語」「ID」「レイヤ情報」「プロテクション（CRC 検査）の有無」「ビットレート情報」「サンプリング周波数」「パッディングビット」「プライベートビット」「モード情報」「モードエクステンション」「コピーの許可」「オリジナルデータかコピーデータか」「エンファシス」という順に記述されている（表１）。詳細は仕様書を参照してほしい。

表１、ヘッダ情報

Header()		
情報	ビット数	意味
Syncword	12bit	同期語
ID	1bit	
Layer	2bit	レイヤ情報
Protection_bit	1bit	CRCの有無
Bitrate_index	4bit	ビットレート
Sampling_frequency	2bit	サンプリング周波数
Padding_bit	1bit	パッディングビットの有無
Private_bit	1bit	
Mode	2bit	モード
Mode_extention	2bit	
copyright	1bit	コピーの許可
original/copy	1bit	オリジナルかコピーか
emphasiss	2bit	

このヘッダの節ではそのフレームの MP3 データがどのようなサンプリング周波数で、1 秒にどれくらいのビットを使っているか、レイヤはいくつか、モード（ステレオとかモノラルとか）はどれでエンコードしているのか等、音のモデルを記している。このヘッダ情報を元に、MP3 データを PCM などの音の波形データにデコードしていく。逆に、エンコードする時はこれらの値を最初に決めておき、それに沿ってエンコードする。

ヘッダ情報の次には CRC 巡回冗長検査のビットが並ぶ。しかし、このビット列はヘッダの「Protection_bit」が 0 の場合のみ出力される。出力された場合、そのフレームにおける巡回冗長検査の値を書き込む。検査方法は CRC-16 を使用する。ビット長は 16 ビット。検査の対象のビットはヘッダ情報の 4 バイト分と、付加情報（ヘッダの後ろの 32 バイト分）である。CRC の検査方法は図２の通りである。この CRC 検査がない場合、予期せぬエラーでビット情報に欠損が出た場合、デコードする際ひずみが出てしまう可能性が高い。CRC があれば、ある程度のエラーは回復できるので、

CRC は付加しておくほうが良い。

CRC 検査ビット列の次には、オーディオデータ（Audio_data()）が続く。この部分で初めてグラニュールとチャネルに触れる。このオーディオデータ節では MP3 データのデコードに必要な情報が記述されている。オーディオデータの最初には「main_data_begin」という 9bit のデータがある。main_data_begin とは、メインデータの開始位置を示す値である。同期語からこの値のバイト分だけ、前のフレームのメインデータ節にこのフレームのメインデータが含まれることになる。実は、MP3 は 1 フレームに入るメインデータの量が、1 フレームに入れられる分より少なかった場合、その次のメインデータを、今のメインデータにの後ろに入れることが可能なのである。こうすることにより、無駄を省き、ファイルサイズをより小さくしている。それを実現するための情報がこの main_data_begin である（図 3）。

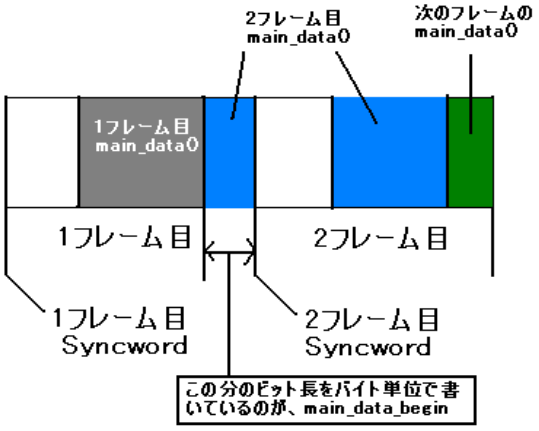


図 3、main_data_begin の説明

Main_data_begin の次は「Private_bit」が続く。この Private_bit は意味を持たない情報である。仕様書によると「将来にわたり、この規格では規定しない」らしい。しかし、この Private_bit はモノラルモードと、それ以外のモードではビット長が変わってしまうので注意が必要だ。モノラルモードでは 5 ビット、それ以外のモードでは 3 ビットと決められている。

このオーディオデータ節の中には、MP3 の特徴とも言えるスケールファクタ（倍率）を定める節も記される。Private_bit の次に来る「Scfsi[ch][scfsi_band]（ch はチャネル、scfsi_band はスケールファクタバンド。同倍率選択情報群。）」がそれである。MP3 で言うスケールファクタとは、音の音量を決める係数である。この値が大きければ大きいほど、音量が大きくなる。Scfsi ではスケールファクタの適用方法を、チャネルごとにスケールファクタバンドで設定している。スケールファクタバンドは、1 チャネルにつき 4 つある。なので、ステレオの場合は 8 つあることになる。短いブロックを使用する場合、Scfsi は 0 とする。

倍率情報が定まったところで、次からはグラニュールとチャネルごとに情報が分かれる。モノラルのデータの場合、2 つのグラニュールに、それぞれ 1 チャネル分のデータが書かれている。それ以外のモードの場合は、2 つのグラニュールにそれぞれ 2 チャネル分のデータが入っている。一番最初の情報は「Part2_3_length[gr][ch]」である。Part2_3_length ではハフマン符号のビット長を記してある 12 ビットの情報である。1 グラニュール中に含まれるハフマン符号の長さはそれぞれ長さが違う。この値を読み、どこまでがどこのグラニュールのどこのチャネルのメインデータかを確認することが出来る。次に続くのが「Big_value[gr][ch]」である。

ビット長は９ビット。Big_value 領域は、ハフマン符号表を用いて０からナイキスト周波数までの全周波数帯域をいくつかの領域に分割し、ハフマン符号化した部分である。この節の Big_value は、その Big_value 領域のデータの個数を表している。Big_value のハフマンコードを符号表と照らし合わせると、x と y の 2 つの値を得ることが出来、その後ろに続くビット列からその 2 値がプラスかマイナスかを知ることが出来る。1 グラニュールにおける Big_value の個数は 576 個と決められている。３つの短いブロックの場合、1 つの窓に最大で 192 個の Big_value を入れることが出来る。

その次は「Global_gain[gr][ch]」である。量子化ステップ情報を対数量子化して付加する付加情報変数である。この値は、逆量子化する時に使用する。次の「Scalefac_compress[gr][ch]」はスケールファクタの同倍率群のビット長を決める情報である。仕様書には、Scalefac_compress 表が付属されている（表２）。同倍率群は Block_type（後述）によって、範囲が定められる。表 2 の Slen1、Slen2 は同倍率群の前半と後半のビット長の値である。Block_type が０，１、又は３のとき Slen1 は同倍率群０～10 に対する倍率のビット数、Slen2 は同倍率群 11~20 に対する倍率のビット数。Block_type が２でかつ Mixed_block_flag（後述）が０のとき、Slen1 は同倍率群０～５に対する倍率のビット数。Slen2 は同倍率群６～１１に対する倍率のビット数。Block_type が２でかつ Mixed_block_flag が１のとき、Slen1 は同倍率群０～７（長い窓）および３～５（短い窓）に対する倍率のビット数。Slen2 は同倍率群６～１１に対する倍率のビット数である。

表 2、Scalefac_compress 表

Scalefac_compress[gr][ch]	slen1	slen2
0	0	0
1	0	1
2	0	2
3	0	3
4	3	0
5	1	1
6	1	2
7	1	3
8	2	1
9	2	2
10	2	3
11	3	1
12	3	2
13	3	3
14	4	2
15	4	3

その次は「Window_switching_flag[gr][ch]」が続く。Window_switching_flag はそのブロックが普通の窓（Block_type==0）以外を使用することを示す 1 ビットの値。Window_switching_flag が設定された時、他のいくつかの変数を特に指定しない限り、次に示す値に設定する。

Region0_count = 7
(Block_type == 1, Block_type == 3 又は Block_type == 2 かつ mixed_block_flag == 1 のとき。)
Region0_count = 8
(Block_type == 2 かつ mixed_block_flag == 1 のとき。)
Region1_count = 36

Big_value 領域の残りの全ての値は、領域１（Region1）に含む。Window_switching_flag が設定されていない（０である）時、

Block_type は 0 とする。Region0_count、Region1_count については、後述する。

Window_switching_flag の次の情報は「Block_type[gr][ch]」である。この情報は、対応するグラニュールのブロックタイプ（窓タイプ）を示す。ブロックタイプには開始ブロック、3 つの短いブロック、停止ブロックの３つと、予約されたブロックタイプがあり、主に 3 つの短いブロックを利用してエンコードする（表３）。開始ブロックは、主に MP3 データの 1 フレーム目（開始フレーム）で使用される。短いブロックは、高い周波数の音をエンコードする際、長いブロックではプリエコーがかかってしまうため、３つに短く切り、時間解像度をよくして、プリエコーの発生を抑えるためにできた。停止ブロックは、最後のフレームで使われる。

表３、Block_type

Block_type[gr][ch]	ブロックタイプ
0	予約
1	開始ブロック
2	3つの短いブロック
3	停止ブロック

Block_type および mixed_block_flag は、そのブロックの数値（情報）の並び方、変換の回数についての情報を与える。Window_switching_flag == 1 のとき、mixed_block_flag は低域側の多重位相フィルタ分割帯域を普通のブロックタイプを使って符号化するかどうかを示す。長いブロックの場合（Block_type が 2 でない場合、または Block_type == 2 であり、Mixed_block_flag == 1 であり、かつ低域側分割帯域の場合）IMDCT は 18 個の入力ごとに 36 個の出力を生成する。その出力に Block_type に応じた窓をかけ、その前半を前ブロックの後半とかさねあわせ処理をする。得られたベクトルを多重位相フィルタバンク合成部の一つの帯域の入力とする。短いブロックの場合（Mixed_block_flag == 1 であり、Block_type == 2 であり、かつ高域側分割帯域の場合、又は mixed_block_flag == 0 でありかつ Block_type == 2 の場合）、それぞれ 12 個の出力値を生成する変換を 3 回実行する。この 3 ベクトルに窓を掛け、互いに重ねあわせ処理をする。得られたベクトルの両端にそれぞれ 6 個の”0”を連結し、長さ３６のベクトルを作り出し、それを長いブロックに用いた変換の出力と同様に処理をする（図４）。Block_type の次は、「Mixed_block_flag[gr][ch]」の情報が入っている。Mixed_block_flag は高域側で使用する窓と異なる窓で低域側を変換することを示す。

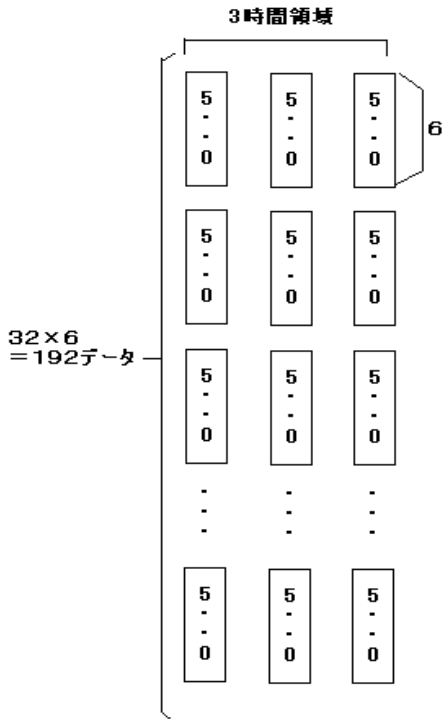


図４、３つの短いブロックの１ブロック

Mixed_block_flag == 0 のとき全てのブロックに Block_type[gr][ch]が示す変換を行う。Mixed_block_flag == 1 のとき、最低域側の 2 個の多重位相分割帯域に対応する周波数成分を普通の窓(Block_type == 0）で変換し、残りの 30 個の分割帯域を Block_type[gr][ch]に応じて変換する。Mixed_block_type が設定された場合、そのグラニュールには短いブロックと長いブロックが混在していることになる。Mixed_block_type の次は「Table_select[gr][ch][region]」が入っている。この情報は、最大量子化値および信号の局所的な統計的性質に応じて異なるハフマン符号表を使用する。合計３２種の符号表があり、この中から適切な符号表を使い符号化する。この Table_select で選択された符号表は、Big_value 領域のみで使用される。MP3 におけるハフマン符号化は、通常のハフマン符号化とは違い、もとからコード表が決まっている。この表の中から音質とファイルサイズを見て適切な表を選択しなければならない。ここの部分で、複雑な計算をする。この部分にかかる時間を短縮することにより、エンコード速度の向上につながると考えている。このコード表には Big_value 領域で得られる x と y の２つの値が記されている。

次に入る情報は、「subblock_gain[gr][ch][window]」である。あるサブブロックについて global_gain[gr][ch]からの偏位量（4 のべき乗による量子化）を示す。Block_type == 2（短い窓）の場合だけ使用する。復号器においてサブブロックの数値を以下の式で除する

$$4^{(\text{subblock\_gain[gr][ch][window]})}$$

Subblock_gain の次には、「region0_count[gr][ch]」が入る。ハフマン符号器の能力を高めるためにスペクトルを更に分割する。これは、Big_value で表される領域を更に再分割することに相当する。この再分割は誤り耐性改善および符号化率向上を目的として行う。領域０、１又は２の３領域を用いる。各領域を最大量子化値および信号の局所的性質に応じて、異なるハフマン符号表で符号化する。region0_count には、count1 領域（後述）の前半の領域の終点の同倍率群番号が書かれている。Count1 領域は、仕様書付属の同倍率群表と、ハフマン符号表 A、およびハフマン符号表 B（後述）を使い変換するため、この値が必要となる。領域の境界はスペクトルの同倍率群の分割点と一致させる必要がある。値、region0_count は領域 0（region0）の同倍率群数より 1 少ない値とする。短いブロックの場合、各同倍率群表は各短い窓に対し、1 回ずつ、合計 3 回数える。したがって、region0_count の値が 8 の時、領域 1 を同倍率群表 3 からはじめることを示す。Block_type == 2 かつ Mixed_block_flag == 1 のとき、そのグラニュールの同倍率群総数は 12× 3 = 36 とする Block_type == 2 かつ Mixed_block_flag == 1 のとき、同倍率群の総数は、８＋９×３＝３５とする。Block_type が２でないとき、同倍率群の総数は２１とする。Region0_count の次には、「Region1_count[gr][ch]」が入っている。Region0 と同じく、Count1 領域のスペクトルの分割点を示している。領域内の同倍率群数より少ない数とする。Block_type == 2 のとき、異なる時間スロットを表す同倍率群は別々に数える。

その次は「preflag[gr][ch]」である。Preflag は量子化値の高域を更に増幅する簡便な方法

である。Preflag を設定すると、仕様書の付属の表に示された値を倍率に加算する。これは、逆量子化した倍率に表の値を乗算することと投下である。Block_type == 2（短いブロック）の時、Preflag は使用しない。

次は、「Scalefac_scale[gr][ch]」である。Scalefac_scale に応じて、倍率は、２又は√２のステップ幅で対数量子化する。次の表に各ステップ幅に対する逆量子化の式で使用する、倍率への乗数を示す（表４）。

表４、scalefac_scale

scalefac_scale[gr][ch]	scalefac_multiplier
0	0.5
1	1

次は、Count1table_select[gr][ch]である。大きさ 1 以下の量子化値からなる四つ組みの領域に対して、使用可能な 2 種のハフマン符号表のうちの一つを選択する。この 2 種ハフマン符号表が付属書の符号表 A、符号表 B である。Count1table_select で、Count1 領域が符号表 A か符合表 B を使っているかが分かる（表５）。

表５、Count1table_select

count1table_select[gr][ch	選択するハフマン符号表
0	付属書B（規定）表 B.7(A)
1	付属書B（規定）表 B.7(B)

これらの情報が、ステレオの場合、1 グラニュールに 2 チャネル分。グラニュールは 1 フレームに 2 つあるので、合計４つ分出力されている。以上、ここまでの情報が Audio_data である。

Audio_data ではデコードに必要な情報が記載されていた。次の情報は、倍率に関する情報の「main_data()」である。仕様書によれば「量子化雑音を周波数領域で重み付けするために、倍率を用いる。量子化雑音の重み付けが正しければ、人間はその量子化雑音を知覚できない。」と書いてある。つまり、MP３の一つの特徴である、人間が聞くことの出来ない高音域を削るということをこの倍率を用いて実現していると言うことである。レイヤ１とレイヤ 2 とは異なり、レイヤ３の倍率は量子化した信号の局所的最大値について、何も規定しない。レイヤ 3 では倍率は複合器で数値の集まりに対する割り算の除数を得るためだけに使用する。レイヤ３の場合、その集まりはいくつかの周波数成分にまたがっている。これらの集まりを同倍率群と呼び、可能な限り臨界帯域に一致するように選択する。

Scalefac_compress 表は、倍率０～１０が０～１５（最大長 4 ビット）の範囲にあること、および、倍率１１～２１は０～７（最大３ビット）の範囲にあることを示す。スケールファクタのビット長は０～4 ビットで、Scalefac_compress 表から得られる Slen1 と Slen2 がそのままビット長になる。スケールファクタの符号化に使用するビット数は Part2_length と呼び、次のように計算する。

・block_type == 0, 1　又は3（長いブロックに対して）

Part2_length = 11 × slen1 + 10 × slen2

・block_type == 2（短いブロック）かつ mixed_block_flag == 0に対して

Part2_lenght = 18 × slen1 + 18 × slen2

・block_type == 2（短いブロック）かつ mixed_block_flag == 1に対して

Part2_length = 17 × slen1 + 18 × slen2

Audio_data の次は、「Huffmancodebit()」つまり、ハフマン符号である。このハフマン符号の中に、音の元となるデータが詰まっている。Huffmancodebit の中には、Big_value 領域と、Count1 領域と、ZERO 領域という３つの領域がある（図５）。Big_value 領域では32個のハフマン符号表のうち1つの表を選んでエンコードされた値が詰まっている。これらの個数は２の倍数である。この領域では x と y の 2 値が得られる。また、2 値のほかに linbits と言う値も得られる。linbits は各は不満符号表に書かれているので、その値の分のビットを読む。Count1 領域では符号表 A か符合表 B でエンコードされた値が入っている。この領域では、v、w、x、y の 4 値が得られる。いずれも、－１，０，１のどれか。個数は４の倍数でなければならない。ZERO 領域は 0 の値が入っている領域である。多くの場合、前のフレームの main_data()が入っているため、ビットストリームには出力されていない。

Big_value 領域は次のパラメタで構成されている。

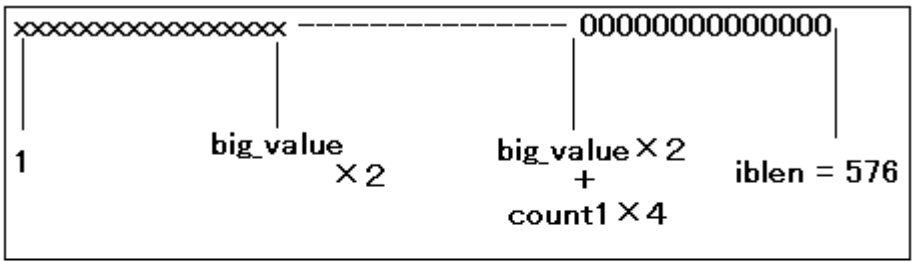
- Hcod [|x|] [|y|]
  ハフマン符号表への引数。この引数を使って、x と y の２値を得ることが出来る。
- hlen [|x|] [|y|]
  値 x、y のハフマン符号長符号表への引数。要は、ハフマン符号のビット長である。
- signx
  もし、x の値がある場合、x の値が正の値か、負の値かを示す。０の場合、正。1 の場合負。
- signy
  もし、y の値がある場合、y の値が正の値か、負の値かを示す。他は signx と同じ。
- linbits
  linbitsx 又は linbitsy がある時、そのビットの長さ。

Count1 領域は次のパラメタで構成されている。

- Hcod [|v|] [|w|] [|x|] [|y|]
  ハフマン符号表への引数。Big_value 領域とは違い、v、w、x、y の 4 値を得ることが出来る。
- signv
  もし、v の値がある場合、v の値が正の値か、負の値かを示す。０の場合、正。1 の場合、負。
- signw
  もし、w の値がある場合、w の値が正の値か、負の値かを示す。他は signv と同じ。

あとは Big_value と同じく、signx、signy がある。意味も同じである。

これらから得られた数値を逆量子化（IMDCT 変換）することにより、音の波形を得ることが出来る。デコードするとき Block_type == 2（短いブロック）の場合、Big_value 領域は同倍率群表を使い、デコードする（図６）。
以上の内容が、MP3 の１フレームのビットストリーム構造になっている。



値 000 は全て0。0の個数は2の倍数
値 --- は-1, 0, 1のいずれか。これらの個数は4の倍数。
値 xxx は制限なし。これらの個数は2の倍数。
iblen は 576。

1フレーム中に576個のデータがある。

図５、Huffmancodebit の中身

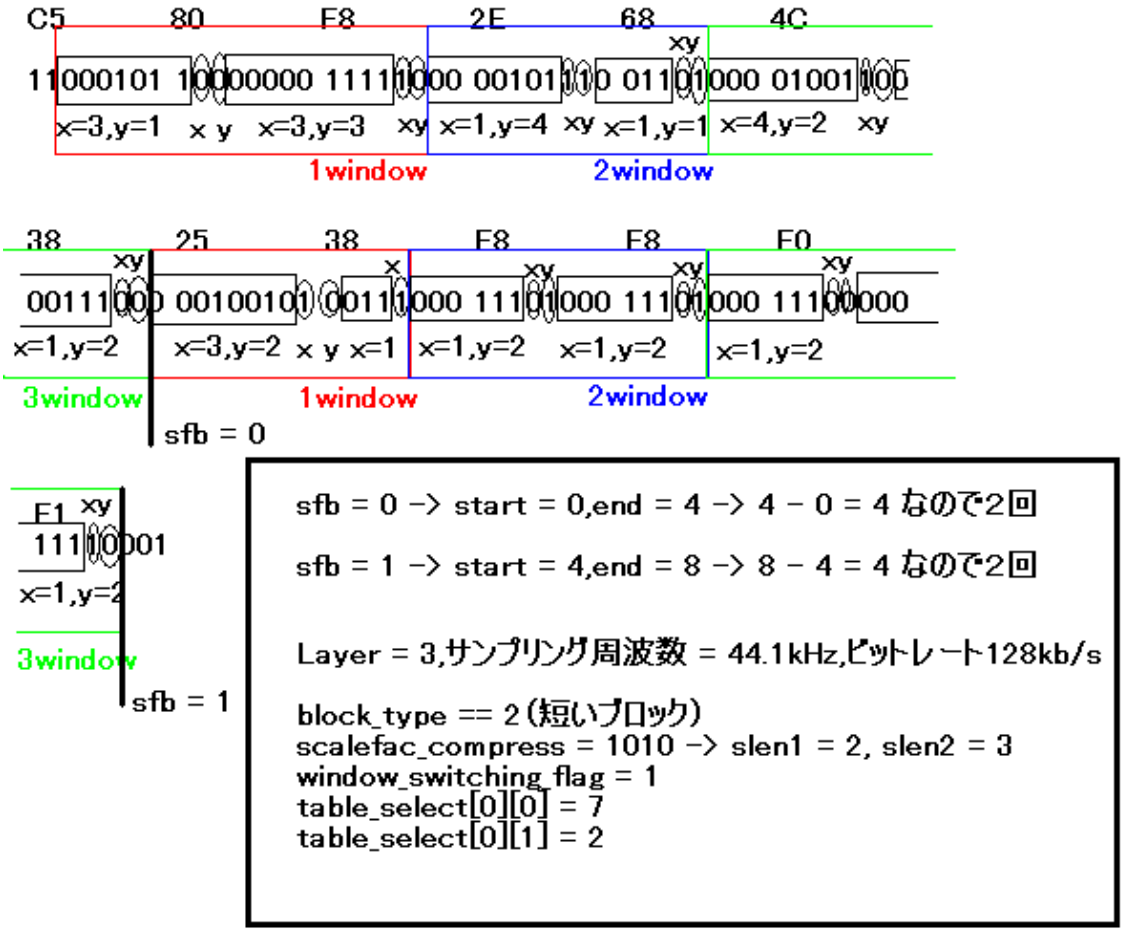
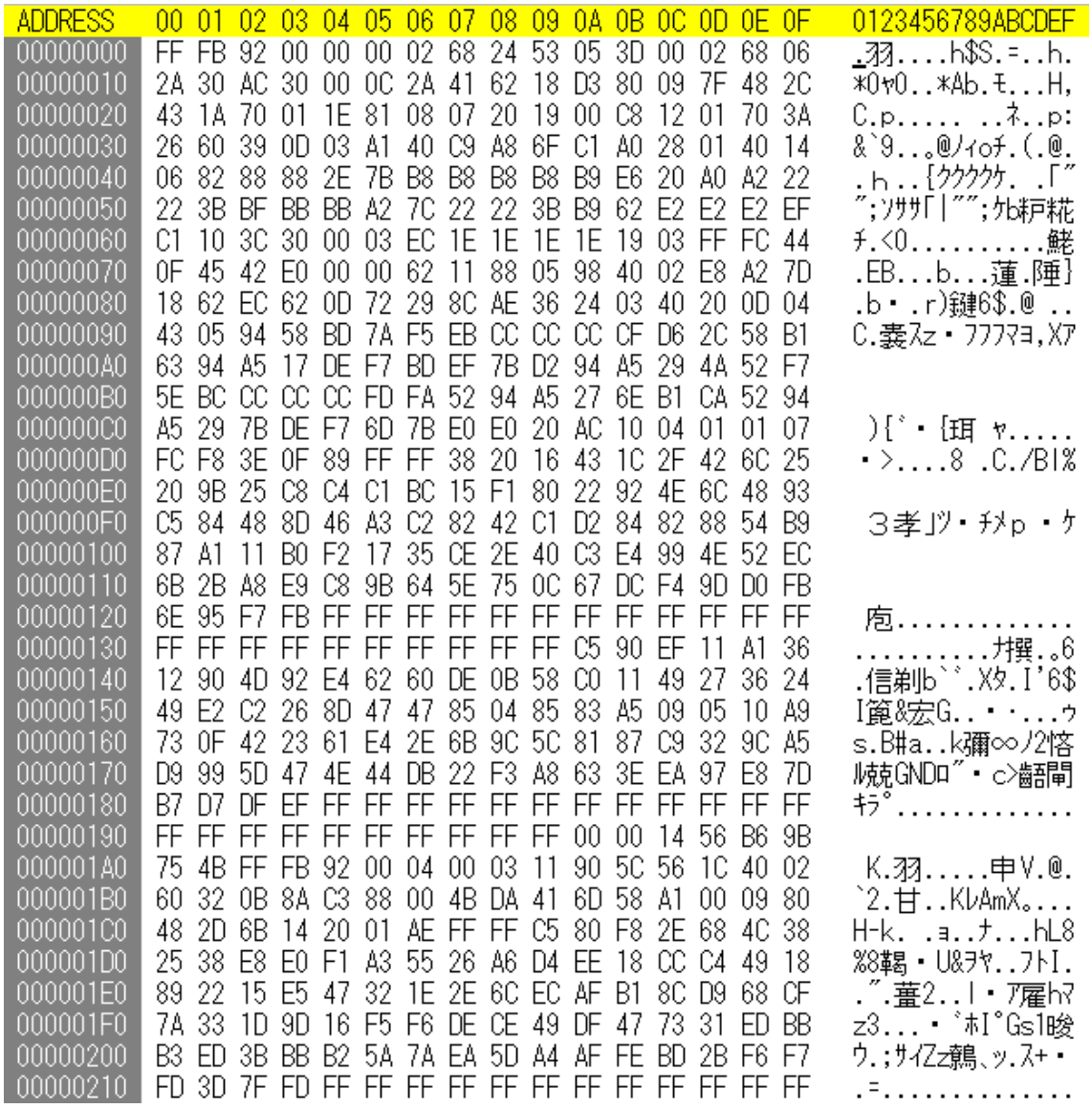


図６、short block におけるデコード

４、MP3 のデコード

ビットストリーム構造が分かったので早速本物の MP3 をデコードしてみよう。まずエンコードする WAVE ファイルを用意する。エンコーダは Dist10 を使用する。ここで忘れてはいけないことがある。Dist10 は Windows で標準とされている PCM や RIFF 形式の WAVE ファイルだと、エンコードできないので、注意が必要だ。Dist10 がエンコードできるファイル形式は AIFF と raw 形式のファイルでなければならない。私は「TiMidity++」というソフトウェアシンセサイザが AIFF ファイルを出力できるので、こちらにて再生時間が短めな MIDI ファイルを AIFF ファイルに出力した。ソースとなるファイルが用意できたので、Dist10 を使い、MP3 にエンコードする。サンプリング周波数は 44.1kHz、ビットレートは 128kb/s、モードはステレオ。プロテクションは使用しないでエンコードする。ここで注意。Dist10 には聴覚心理モデルが２つある（正しくはＭＰ３エンコーダ。ISO の仕様書にも、同様に２つの聴覚真理モデルの説明が記載されている）。聴覚心理モデル１は、Layer３には使えない。Dist10 は、デフォルトで使用する聴覚心理モデルは「聴覚心理モデル１」なので、エンコード時にオプションで「聴覚心理モデル２」に指定しなければならない。このように細かくオプションを決めてあげないと、きちんとエンコードできない。出力される MP3 ファイルの拡張子は最初「～.mpg」になっている。エンコードし終わったら、一度再生して音を聞いてみよう。成功していれば、ソースのファイルとほぼ同じ音が聞こえるはずだ。しかし、失敗していると、変なノイズがのっていたり、「ザー」というノイズしか乗っていない場合がある。再生するソフトウェアは Windows 環境では「SCMPX」がお勧めである。SCMPX は、古くからフリーソフトとして配布されている MP3 を再生できるソフトウェアである。エンコードも可能である。LAME エンコーダとは違い、ある程度 ISO 仕様書にそった MP３ファイルを生成するようである。自前のエンコーダとデコーダを持っており、CD から直接 MP3 にエンコードすることが、CD2WAVE というソフトウェアの外部ツールとして使用することで、可能だった。何故、私が SCMPX を奨めるかと言うと、実は Dist10 でエンコードした MP3 ファイルを Windows Media Player で再生しようとしたところ、どれも再生できずにフリーズしてしまった。（私のミスかもしれない。）しかし、RIFF 形式の WAVE ファイルをエンコードした MP3 ファイルは再生できた。けれども雑音しか入っていなかった。他の知っている MP3 を再生できるプレイヤーソフトも使ってみたが、やはり雑音だけだったり、再生出来ずに終了した。MP3 ファイルが壊れているのかもしれないと思い、他の音源もエンコードしたものの、結果は同じだった。しかし、SCMPX ならばとりあえずは再生可能だったので、これはきっと MP3 ファイルなのだろう、と仮定して作業を進めた（教授曰く、「Windows Media Player が ISO の仕様を満たしていない可能性も否定できない。」そういえば昔、Windows98 に付属していた Media Player か、レコーダだったかに、MP3 エンコード機能がついていたのを見つけて、期待に胸を膨らませながら、僕が好きだったある曲をエンコードしたら、プツプツというノイズと、ブロックノイズだらけで、がっくりした記憶がある・・・。ちなみに、サンプリング周波数は 44.1kHz、ビットレー

ト 128kbps、ステレオ。記憶があいまいだけれども、とにかくノイズだらけだったのは記憶に残っている)。エンコードが終わり、音も確認し終わったら、次はバイナリエディタで MP3 ファイルを開く。Windows 環境ならば「Stirling」、Linux 環境ならば「bvi」がお勧めである。私は現在 Windows 環境であるので Stirling を使用する。さて、バイナリエディタで先ほど作成した MP3 ファイルを開くと図 7 のようになる。これだけを見ても何がなんだか分からないはず。これを見ただけで分かったら、あなたには才能があるのかもしれない（Dist10 以外のエンコーダでエンコードすると、ID3 タグという情報タグが一番上、もしくは一番下についている可能性が高い。すると、この部分には生の文字が入っているため、バイナリエディタの右の文字列表示に、目で読める部分があったりする。Dist10 には ID3 タグがない。なので、仕様書と照らし合わせて解読するには、非常に良い)。まず、バイナリエディタの見かたを説明していく。一番上の黄色い部分は、列数を意味している。バイナリエディタは 16 進数で表示されるので、最後の「0F」は１６列目ということになる。左の灰色の部分は、白い部分に書かれた数値のアドレス番号を表している。なので、2 行目の 1 番最初のアドレスは１７なので、F（１６）から 1 繰り上がって「１０」となる。さて、このバイナリエディタを使って、MP3 をデコードしていきたいと思う。MP3 の一番最初は「同期語（syncword)」の 1 が 12 ビット並んでいるであるから、１６進数では「FFF」となるはずである。バイナリエディタで開いた MP3 の一番最初は「FFF」である。よってここがヘッダの始点となる。ヘッダは始点から 4 バイト分であるので、0 から 4 番までのアドレスを読む。バイナリコードでは「FF FB 92 00」。これを 2 進数 1 ビット単位に直すと「1111 1111 1111 1011 1001 0010 0000 0000」となる。これをデコードする。最初に述べた 12 ビットは同期語である。



ADDRESS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000	FF	FB	92	00	00	00	02	68	24	53	05	3D	00	02	68	06	..羽....h$S.=..h.
00000010	2A	30	AC	30	00	0C	2A	41	62	18	D3	80	09	7F	48	2C	*0ヤ0..*Ab.モ...H,
00000020	43	1A	70	01	1E	81	08	07	20	19	00	C8	12	01	70	3A	C.p..... ..ネ..p:
00000030	26	60	39	0D	03	A1	40	C9	A8	6F	C1	A0	28	01	40	14	&`9...@ﾉｨoﾁ.(.@.
00000040	06	82	88	88	2E	7B	B8	B8	B8	B8	B9	E6	20	A0	A2	22	.h..{ｸｸｸｸｹ. .「"
00000050	22	3B	BF	BB	BB	A2	7C	22	22	3B	B9	62	E2	E2	E2	EF	";ｿｻｻ「|""";ｹb粁糀
00000060	C1	10	3C	30	00	03	EC	1E	1E	1E	1E	19	03	FF	FC	44	ﾁ.<0.....鮭
00000070	0F	45	42	E0	00	00	62	11	88	05	98	40	02	E8	A2	7D	.EB...b...蓮.陲}
00000080	18	62	EC	62	0D	72	29	8C	AE	36	24	03	40	20	0D	04	.b・.r)鍵6$.@ ..
00000090	43	05	94	58	BD	7A	F5	EB	CC	CC	CC	CF	D6	2C	58	B1	C.轰ｽz・ﾌﾌﾌﾏﾖ,Xｱ
000000A0	63	94	A5	17	DE	F7	BD	EF	7B	D2	94	A5	29	4A	52	F7	
000000B0	5E	BC	CC	CC	CC	FD	FA	52	94	A5	27	6E	B1	CA	52	94	
000000C0	A5	29	7B	DE	F7	6D	7B	E0	E0	20	AC	10	04	01	01	07	)[ﾞ・[珥 ャ.....
000000D0	FC	F8	3E	0F	89	FF	FF	38	20	16	43	1C	2F	42	6C	25	・>....8 .C./Bl%
000000E0	20	9B	25	C8	C4	C1	BC	15	F1	80	22	92	4E	6C	48	93	
000000F0	C5	84	48	8D	46	A3	C2	82	42	C1	D2	84	82	88	54	B9	３孝」ｿ・ﾁﾒp ・ｹ
00000100	87	A1	11	B0	F2	17	35	CE	2E	40	C3	E4	99	4E	52	EC	
00000110	6B	2B	A8	E9	C8	9B	64	5E	75	0C	67	DC	F4	9D	D0	FB	
00000120	6E	95	F7	FB	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	庖.....
00000130	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	C5	90	EF	11	A1	36	.....ﾅ撰..6
00000140	12	90	4D	92	E4	62	60	DE	0B	58	C0	11	49	27	36	24	.信弟b``.Xﾀ.I'6$
00000150	49	E2	C2	26	8D	47	47	85	04	85	83	A5	09	05	10	A9	I篦&宏G..・・...ｳ
00000160	73	0F	42	23	61	E4	2E	6B	9C	5C	81	87	C9	32	9C	A5	s.B#a..k彌∞ﾉ2悟
00000170	D9	99	5D	47	4E	44	DB	22	F3	A8	63	3E	EA	97	E8	7D	ﾙ兢GNDﾛ"・c>齟閘
00000180	B7	D7	DF	EF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ｷﾗ°.....
00000190	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	00	14	56	B6	9B	
000001A0	75	4B	FF	FB	92	00	04	00	03	11	90	5C	56	1C	40	02	K.羽.....申V.@.
000001B0	60	32	0B	8A	C3	88	00	4B	DA	41	6D	58	A1	00	09	80	`2.甘..KﾚAmX｡....
000001C0	48	2D	6B	14	20	01	AE	FF	FF	C5	80	F8	2E	68	4C	38	H-k. .ｮ..ﾅ...hL8
000001D0	25	38	E8	E0	F1	A3	55	26	A6	D4	EE	18	CC	C4	49	18	%8鞳・U&ヲﾔ..ﾌﾄI.
000001E0	89	22	15	E5	47	32	1E	2E	6C	EC	AF	B1	8C	D9	68	CF	."蕙2..l・ｱ雇hﾏ
000001F0	7A	33	1D	9D	16	F5	F6	DE	CE	49	DF	47	73	31	ED	BB	z3...・ﾞﾎI°Gs1ｂ俊
00000200	B3	ED	3B	BB	B2	5A	7A	EA	5D	A4	AF	FE	BD	2B	F6	F7	ウ.;ｻｲZz鶫、ｯ.ｽ+・
00000210	FD	3D	7F	FD	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.=.....

図 7、バイナリエディタで開いた MP3 データ

ここからは、ISO/IEC11172-3 つまり MP3 の ISO 仕様書を読みながら説明する。もし手元に ISO 仕様書がある場合は、見て確認しながら読むとより理解できるはずだ。さて、解読作業に入る。同期語の次のビットは「ID」を示している。この ID が１であれば、MP3 である。2 進数コードは「１」なので、MP3 である。次は「レイヤ」だ。レイヤは２ビットなので２ビット分読むと「０１」だ。レイヤに０１が入っている場合、レイヤは３になるので、このデータはレイヤ３でエンコードされていることが分かる。次は Protection_bit の１ビットだ。値は「１」が入っているので、このフレームのデータには CRC 巡回冗長検査が出力されていない。次は bitrate_index の 4 ビットだ。「1001」が入っている。ISO 仕様書の表を見ると、レイヤ３の 1001 は、128kbit/s なので、このデータは一秒に 128ｋビットのビットレートになっている。次は sampling_frequency の 2 ビット。値は「００」が入っている。仕様書の表では sampling_frequency が「00」のときは標本化周波数（サンプリング周波数）は 44.1kHz になっている。次は、モードだ。2 ビット。値は「00」仕様書によれば、００はステレオである。次は mode_extention の 2 ビット。こちらは、「００」である。残りの４ビットは copyright の１ビット、original/copy の１ビット、emphasis の 2 ビットである。全て０なので著作権なし、複製されたデータ、エンファシスはなし。となっている。最初に設定したとおりの MP3 データが出力されているのが分かる。Protection_bit は１なので、CRC データはない。次は、Audio_data() だ。Audio_data はモノラルモード以外では、ヘッダの4倍と以降の32バイト分のデータがこれに相当するのでそれを読む。全て読むと分かりづらくなるので、０グラニュール０チャネルのみ読む。そこからあとは、図８、図９、図１０、図１１に記す。Audio_data()の最初は main_data_begin の 9 ビットである。このフレームは、最初のフレームなので、main_data_begin は０になっている。Private_bit はステレオなので３ビットあればいい。次の Scfsi はステレオモードなので 2 チャネル分取らなければならない。Scfsi 一つ分で１ビット、1 チャネルに４つの scfsi_band があるので合計８ビット。Scfsi の次は、part2_3_length にはいる。こちらは 12 ビットである。値は「001001101000」だ。これを 10 進数に直すと 616 になる。よってハフマン符号のビット数は 616 ビットである。次は Big_value の 9 ビット。値は「001001000」。10 進数に直すと７２．よって、72 個の Big_value が huffmancodebit の big_value 領域に入っていることになる。次は、Global_gain の８ビットだ。そして、scalefac_compress の 4 ビットが続く。Scalefac_compress は「0000」の値が入っているので slen1 と slen2 は両方とも０．このフレームは音が出ないと予測される。次は window_siwtching_flag の１ビット。値は１．Window_switching_flag が設定されているので、次は block_type の 2 ビットになる。こちらは、「01」が入っているので開始ブロックであることが分かる。次は、mixed_block_flag の１ビット。「０」が入っている。その次の table_select は、window_switching_flag が１なので領域は２つ。領域１の table_select は 01111、領域２の table_select は 01000。なので、領域１ではハフマン符号表１５、領域２ではハフマン符号表８を使う。

Subblock_gain は 3 つの窓分用意している。１つの窓で 4 ビット使うので、12 ビット分用意されている。このフレームではどれも 0 が入っている。次は Preflag、scalefac_scale、count1table_select の 3 ビット。Preflag は 0、scalefac_scale も 0 . Count1table_select は 1 が入っている。Count1table_select が 1 なので、count1 領域では、符号表 B を使う。他の情報は図 8 から図１１を参照。これらの情報をもとに、ハフマン符号をデコードする。０グラニュール、０チャネル目のハフマン符号は、開始ブロックなので、長いブロックである。よって Huffmancodebit()は big_value と count1 領域、０領域で構成されている。今、Big_value 領域はさらに二つの領域に分けることが出来る。前半の領域は Region0 で、後半の領域は Region1 だ。Table_select で選択された符号表は、符号表１５と符号表８なので、Region0 は符号表１５、Region１は符号表８でデコードする。ここまでのデータが、シングルチャネル（モノラル）の場合、1 グラニュール、1 チャネルの２つ、それ以外のモード（ステレオ等）の場合、2 グラニュール 2 チャネルの４つあれば、そのあとに続くデータは、MP3 のメインデータ（量子化された値）なので、とりあえず ISO で定められた仕様を満たしたデータになる。

さて、Audio 情報を読みどういう音の情報が、どれくらいの音質で入っているかがわかったので、今度は音そのものを符号化圧縮した Main_data（）と Huffmancodebit（）を解読していく。私、実はこの項につい正しい答えが出せていない。理由は、エンコーダもデコーダも作れず、これであっているのかどうか、確かめられなかったからである。唯一、私が確かめられるエンコーダとデコーダは、Dist10 だけだった。なので、この項は Dist10 の仕様書と Dist10 のソースコードを頼りに、私なりに解釈したデコード方法である。よって確証性が無い。

さて、本題に入る。まず Audio_data 節と Huffmancodebit 節は図１２のようなになっていることを確認してほしい。こちらは、仕様書にも同じような図があるので、正しいはずである。解釈が間違っていなければのはなしではあるが。モノラルの場合、big_value 領域には 1 グラニュールにつき 1 チャネル分のデータがあり、それ以外（ステレオなどの 2 チャネル必要とする）の場合、1 グラニュールにつき 2 チャネル分ある。また、倍率に対応する２つの領域に区切られており、前半の領域を region0 領域、後半の領域を region1 領域と呼ぶことにする。Count1 領域は、「-1,0,1」のいずれかの値が入っている領域で、「v,w,x,y」の 4 値得ることが出来る。最後の赤く囲まれた部分は、次のフレームのデータで、次のフレームの main_data_begin に入っている数のバイト数分のビットが入っている（つまり main_data_begin × ８ビット入っている）。１つのグラニュール分に big_value 領域と count1 領域、そして出力されない 0 領域を含めて 576 データある。Big_value 領域と count1 領域を合わせて 576 データに満たない場合、0 領域がある。0 領域を出力しないことで、ファイルサイズを小さくしている。

まず、main_data 節の解読に入りたい。この節では、倍率の細かい指定をしている。倍率は scalefac_l（長いブロック用）と scalefac_s（短いブロック用）の二種類があり各ブロックタイプに応じて使い分ける。倍率のビット長は scalefac_compress 表を使い、図１４の式で求めることが出来る。倍率のビ

ット長を part2_length と呼びこの値は、ブロックタイプと scalefac_compress に応じて変化する。Part2_3_length とは違うものなので注意。この part2_length の分だけ倍率が 1 グラニュールの 1 チャネル、あるということになる。さて、図１４の式を使いこのフレームの倍率を求める。０グラニュール、0 チャネルの scalefac_compress には０が入っている。Slen1 は０、slen2 は０。よって、part2_lenght は０。０グラニュール 0 チャネルの倍率は出力されていないことになる。同様に、0 グラニュール 1 チャネル、1 グラニュール０チャネル、1 グラニュール 1 チャネルも scalefac_compress が０なので、このフレームのビットストリーム上には倍率に関するビットは出力されていない。

さて、big_value 領域の解読に入る。Big_value 領域を解読するには、仕様書付属のハフマンコード表を利用して解読していくのは前述したとおりである。Big_value 領域に適応できる符号表は 0-31 の符号表である。では、0 グラニュールの table_select はどうなっているかと言うと、「table_select[0][0][0] 01111」「table_select[0][0][1]　01000」なので region0 領域には符号表 15 を、regio1 領域には符号表８を適応してデコードする。0 グラニュール０チャネル目の part2_3_length は 616、big_value は７２が入っているので、0 グラニュール０チャネル目の huffmancodebit 節は 616 ビットあり、そのうち big_value 領域は 72 組ある。その後ろに count1 領域、0 領域と続く。big_value 領域は図１３のような構造になっている。符号表に書かれたハフマン符号と、MP3 のビットストリーム上に書かれているコードとを照らし合わせて、デコードしていく。ハフマン符号表とビットストリームから得られたハフマン符号から、x,y の 2 値を取り出すことができ、ハフマン符号の後に続く 2 つの値が x,y が正の値か負の値かを示している。0 グラニュール 0 チャネル目はブロックタイプが開始ブロックなので長いブロックである。そして region0_count が７、region1_count は３６なので、仕様書付属の同倍率群表を使い、big_value 領域を region0 領域と region1 領域で分割する。72 組データが取れた時点で、big_value 領域は終了する。そして、count1 領域に入る。Count1 領域では符号表 A と符号表 B のどちらかを選択し符合する。どちらの符号表を使えば良いかは、count1table_select に書いてあるので、それをみて判断する。0 グラニュール 0 チャネル目の count1table_select は 1 が入っているので、符号表 B を使いデコードする。616 ビット目で、count1 領域が終わり、0 グラニュール 0 チャネルのハフマン符号ビットのデコードが完了する。このやり方を残りの３つにも適応すると、1 フレームのハフマン符号のデコードが完了する。

上記から得た値を逆量子化器に入れ、得られた値を合成フィルタバンクに入力し、並べ替えする。逆量子化に使用する公式は図１５に示す。このとき isi はバッファ指標 i におけるハフマン複合値とし、合成フィルタバンクへの入力値を xri とする。図１５の pretab はプリエンファシスのことである。210 と言う値はシステム定数で、出力値を適切に倍率変換するために用いる。短いブロックを使用した場合、逆倍率変換したデータ xr[scf_band][window][freq_line] は IMDCT 演算の前に分割帯域順 xr[subband][window][freq_line] に並べ替えなければならない。

逆量子化が終わると、次はステレオ処理が入る。ステレオ処理では、和差ステレオモード（ミッドサイドステレオモード)、共包絡ステレオモード（ジョイントステレオモード）の 2 種類を決定する。和差ステレオモードにおいて、正規化された和チャネルの Mi と Si の値を左右チャネル値 Li と Ri の変わりに伝送する。Li と Ri は次の式で復元する。

$$Li = \frac{Mi + Si}{\sqrt{2}}$$

$$Si = \frac{Mi - Si}{\sqrt{2}}$$

共包絡ステレオは音声圧縮規格が、普通のステレオと、和差ステレオを、ブロックごとに分けて音声圧縮をかける時の方法のことで、この方法でエンコードすると、和差ステレオだけでエンコードするよりも、効率がよく圧縮できる。MP3 では共包絡ステレオで符号化された各同倍率群は sb（スケールファクタ・バンド）にたいして次の（1）～（5）を実行する。

（1） 共包絡ステレオ位置 $is_pos_{sb}[sfb]$を右チャネルの倍率から読む。

（2） もしくは、$is\_pos_{sb} == 7$ なら（3）～（5）は実行しない。（is_pos_{sb} は無効）

（3） $$is\_ratio = \tan\left(is\_pos_{sb} \times \frac{\pi}{12}\right)$$

（4） 現在の同倍率群 sb 内の全ての i に対して、次の式に示す計算を行う。

$$Li = Li \times \frac{is\_ratio}{1 + is\_ratio}$$

（5） 現在の同倍率群 sb 内のすべての i に対して、次の式に示す計算を行う。

$$Ri = Li \times \frac{1}{1 + is\_ratio}$$

ステレオ相関の次は、折り返しひずみ削減である。ロングブロックに適応する処理である。ロングブロックの場合、ショートブロックよりも長い時間の音が入っているため、折り返しひずみという雑音が入りやすいので、ここでノイズを取り除き、PCM 標本に悪影響を与えないようにしている。ショートブロックには適応しない。

折り返しひずみ削減の次は、IMDCT を使い、PCM 標本を作る。IMDCT は逆修正離散コサイン変換と言い、エンコーダで使われている MDCT 処理の逆をする処理である。IMDCT の式を図１６に示す。n は窓掛けされた標本値の個数とする（短いブロックに対する n は１２で、長いブロックに対する n は 36 とする。）短いブロックの場合は、3 個のブロックをそれぞれ別々に変換する。長いブロックの場合はそのブロックごと変換する。IMDCT で掛ける窓はブロックタイプに応じて異なる窓を掛ける。各ブロックタイプに対応する窓の形は図１７、図１８の通りである。そして、36 個の値を持つブロックの前半分を直前ブロックの後半分に重ね合わせて加算する。現ブロックの後半分は次のブロックで使用するために保存する。重ねあわせ加算された出力値は３２の多重位相分割帯域ごとに 18 個の時間標本値とする。最初の時間標本値から順に０～１７の番号をつけ、分割帯域も０～３１の番号を付ける。多重位相フィルタバンクに入力する前に全ての奇数分割帯域の奇数時間標本値に-1 を乗じる。

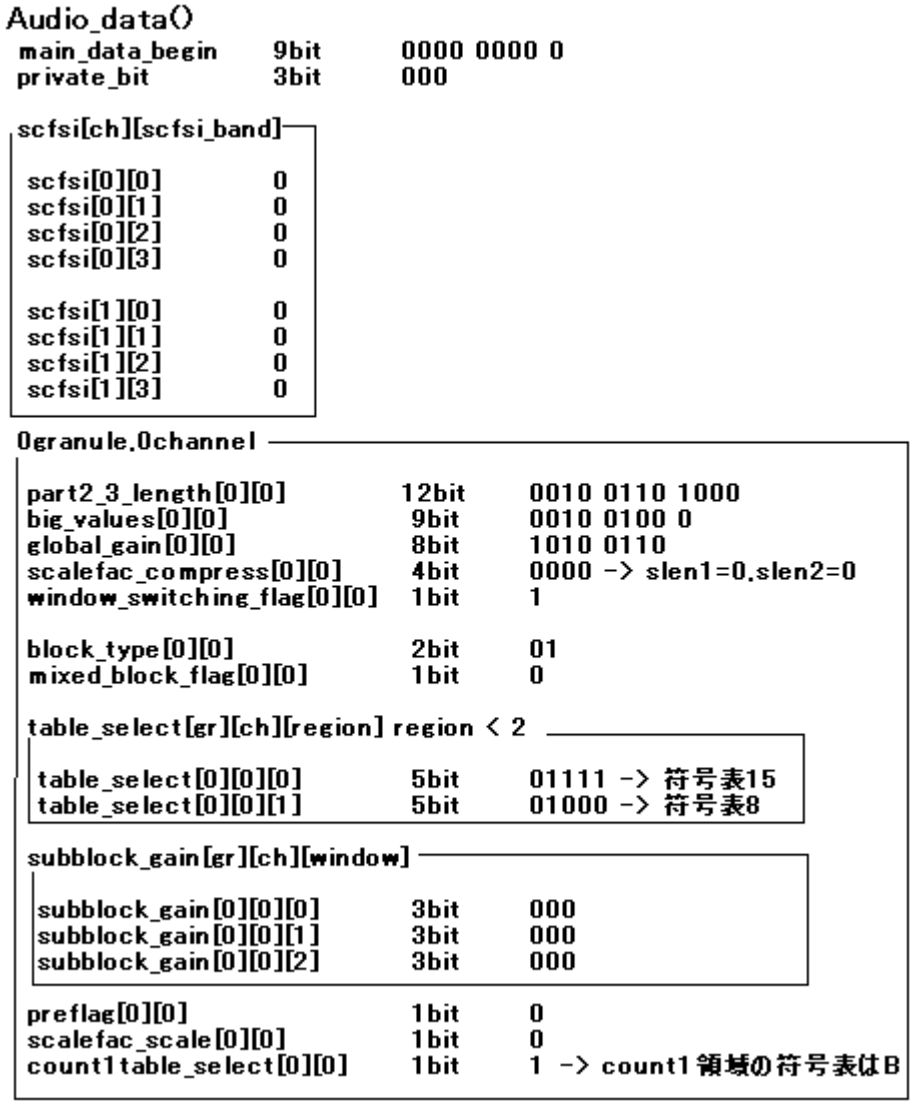


図８、Audio_data その１

0granule,1channel

part2_3_length[0][1]	12bit	0011 0100 0000 -> 832bit
big_value[0][1]	9bit	0011 0001 0 -> 98個
glbal_gain[0][1]	8bit	1010 0011
scalefac_compress[0][1]	4bit	0000 -> slen1=0,slen2=0
window_switching_flag[0][1]	1bit	1
block_type[0][1]	2bit	01
mixed_block_flag[0][1]	1bit	0
table_select[gr][ch][region] region < 2		
table_select[0][1][0]	5bit	11000 -> 符号表24
table_select[0][1][1]	5bit	01100 -> 符号表12
subblock_gain[gr][ch][window] window < 3		
subblock_gain[0][1][0]	3bit	000
subblock_gain[0][1][1]	3bit	000
subblock_gain[0][1][2]	3bit	000
preflag[0][1]	1bit	0
scalefac_scale[0][1]	1bit	0
count1table_select[0][1]	1bit	0 -> count1領域符号表はA

図９、Audio_data その２

1granule,0channel

part2_3_length[1][0]	12bit	0011 0000 1010 -> 778bit
big_value[1][0]	9bit	1001 0000 0 -> 288個
glbal_gain[1][0]	8bit	1011 0001
scalefac_compress[1][0]	4bit	0000 -> slen1=0,slen2=0
window_switching_flag[1][0]	1bit	1
block_type[1][0]	2bit	10 -> 3つの短いブロック
mixed_block_flag[1][0]	1bit	0
table_select[gr][ch][region] region < 2		
table_select[1][0][0]	5bit	01101 -> 符号表13
table_select[1][0][1]	5bit	00111 -> 符号表7
subblock_gain[gr][ch][window] window < 3		
subblock_gain[1][0][0]	3bit	000
subblock_gain[1][0][1]	3bit	000
subblock_gain[1][0][2]	3bit	000
preflag[1][0]	1bit	0
scalefac_scale[1][0]	1bit	0
count1table_select[1][0]	1bit	1 -> count1領域符号表はB

図１０、Audio_data その３

1granule,0channel

part2_3_length[1][1]	12bit	0010 1111 1110 -> 766bit
big_value[1][1]	9bit	1001 0000 0 -> 288個
glbal_gain[1][1]	8bit	1011 0001
scalefac_compress[1][1]	4bit	0000 -> slen1=0,slen2=0
window_switching_flag[1][1]	1bit	1
block_type[1][1]	2bit	10 -> 3つの短いブロック
mixed_block_flag[1][1]	1bit	0
table_select[gr][ch][region] region < 2		
table_select[1][1][0]	5bit	01101 -> 符号表13
table_select[1][1][1]	5bit	00111 -> 符号表7
subblock_gain[gr][ch][window] window < 3		
subblock_gain[1][1][0]	3bit	000
subblock_gain[1][1][1]	3bit	000
subblock_gain[1][1][2]	3bit	000
preflag[1][1]	1bit	0
scalefac_scale[1][1]	1bit	0
count1table_select[1][1]	1bit	1 -> count1領域符号表はB

図１１、Audio_data その４

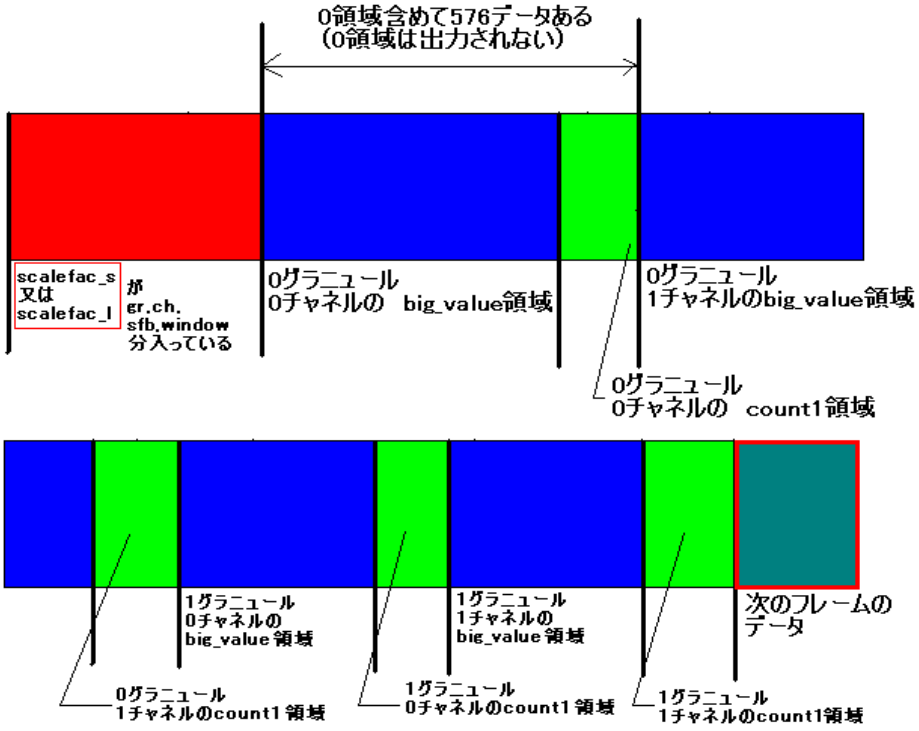


図１２、main_data と Huffmancodebit のビットストリーム

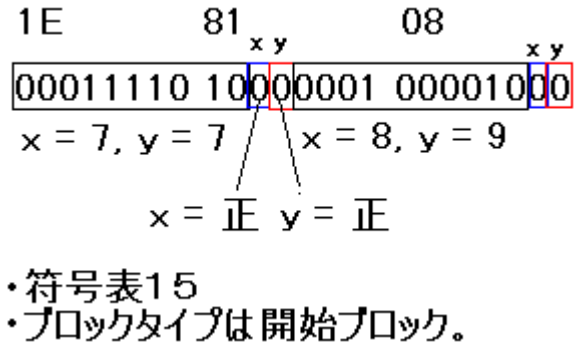


図１３、big_value 領域の構造（long_block）

・block_type == 0, 1 又は3(長いブロック)に対して

part2_length = 11 × slen1 + 10 × slen2 -> 21個のデータ

・block_type == 2(短いブロック)かつmixed_block_flag == 0に対して

part2_length = 18 × slen1 + 18 × slen2 -> 36個のデータ

・block_type == 2(短いブロック)かつmixed_block_flag == 1に対して

part2_length = 17 × slen1 + 18 × slen2 -> 35個のデータ

図１４、part2_length

短いブロックの逆量子化公式

$$xr_i = \mathrm{Sign}(is_i) \times |is_i|^{\frac{4}{3}} \times 2^{\frac{1}{4}(\mathrm{global\_gain[gr][ch][window]} - 210 - 8 \times \mathrm{subblock\_gain[gr][ch][window]})} \times 2^{-(\mathrm{scalefac\_multiplier} \times \mathrm{scalefac\_s[gr][ch][sfb][window]})}$$

長いブロックの逆量子化公式

$$xr_i = \mathrm{Sign}(is_i) \times |is_i|^{\frac{4}{3}} \times 2^{\frac{1}{4}(\mathrm{global\_gain[gr][ch][window]} - 210)} \times 2^{-(\mathrm{scalefac\_multiplier} \times (\mathrm{scalefac\_l[sfb][ch][gr]} + \mathrm{preflag[gr]} \times \mathrm{pretab[sfb]}))}$$

図１５、逆量子化公式

$$Xi = \sum_{k=0}^{\frac{n}{2}-1} Xk \cos\left(\frac{\pi}{2n}\left(2i+1+\frac{n}{2}\right)(2k+1)\right) \quad i = 0, \cdots, n-1$$

図１６、IMDCT 式

・ロングブロック

(a)block_type = 0(普通のブロック)

$$z_i = x_i \sin\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) \quad i = 0, \cdots, 35$$

$$= x_i \sin\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) \quad i = -, \cdots, 35$$

(b)block_type = 1(開始ブロック)

$$z_i \begin{cases} x_i \sin\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) & i = 0, \cdots, 17 \\ x_i & i = 18, \cdots, 23 \\ x_i \sin\left(\frac{\pi}{12}\left(i-18+\frac{1}{2}\right)\right) & i = 24, \cdots, 29 \\ 0 & i = 30, \cdots, 35 \end{cases}$$

(c)block_type = 3(停止ブロック)

$$z_i \begin{cases} 0 & i = 0, \cdots, 5 \\ x_i \sin\left(\frac{\pi}{12}\left(i-6+\frac{1}{2}\right)\right) & i = 6, \cdots, 11 \\ x_i & i = 12, \cdots, 17 \\ x_i \sin\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) & i = 18, \cdots, 35 \end{cases}$$

図１７、ロングブロックの窓

(d)block_type = 2(短いブロック)

3個の短いブロックは別々に窓掛けする。

$$y_i^{(j)} = x_i^{(j)} \sin\left(\frac{\pi}{12}\left(i + \frac{1}{2}\right)\right) \quad \begin{array}{l} i = 0, \cdots, 11, \\ j = 0, \cdots, 2 \end{array}$$

窓掛けされた短いブロックは重ねあわせ加算し
連結しなければならない。

$$z_i \begin{cases} 0 & i = 0, \cdots, 5 \\ \underset{y_i - 6}{(1)} & i = 6, \cdots, 11 \\ \underset{y_i - 6 \quad y_i - 12}{(2) + (2)} & i = 12, \cdots, 17 \\ \underset{y_i - 12 \; y_i - 18}{(2) + (3)} & i = 18, \cdots, 23 \\ \underset{y_i - 18}{(3)} & i = 24, \cdots, 29 \\ 0 & i = 30, \cdots, 35 \end{cases}$$

図１８、ショートブロックの窓

５、MP3 のエンコード方法

次に、エンコードの方法を説明する。MP3 は仕様でエンコード方法が定められていないので、自由にエンコードすることが出来る。エンコードの流れは、大まかに言うと１、量子化、２、ハフマン符号化、３、付加情報出力、４ビットストリームに出力。この手順でエンコードできるはずである。この中でも、一番計算が大変なのが２のハフマン符号化で、32 個ある符号表から、音質とファイルサイズの両方を見て、最適な符号表を選び出さなければならない。エンコードの方法について、仕様書では「1 例」としてエンコードの方法の説明が載っているその説明は以下の通りである。

MP3 は聴覚心理アルゴリズムを用いている。聴覚心理アルゴリズムの主要部分は図１４に示す。

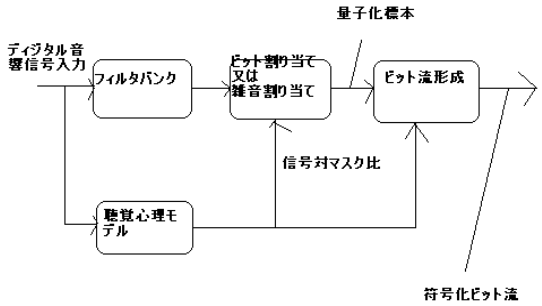


図１４、MP3 の符号化方法

フィルタバンクでは多重位相フィルタバンクおよび混成多重位相フィルタバンク（MDCT フィルタバンク）で各々の時間および周波数上での特殊な写像を行う。フィルタバンクから出力した標本を量子化する。また、このフィルタバンクはレイヤ３の場合、入力信号に依存した 6×３２個または１８×３２個の周波数帯域の解像度を持つ。

聴覚心理モデルは MP3 の場合、モデル２を当てる。最終出力は、分割帯域のグループに対する信号対マスク比（SMR）とする。

ビット割り当て又は雑音割り当てでは、フィルタバンクの出力標本および SMR を考慮しながら、雑音割り当てしていく。

ビット流形成では、量子化されたフィルタバンクの出力、雑音割り当て、およびほかの必要な付加情報の符号化およびフォーマットを行う。レイヤ３の場合はハフマン符号も行う。ビット割り当てまたは雑音割り当ての方法は、レイヤ１およびレイヤ２ではビット割り当て処理として、レイヤ３では雑音割り当てループとして実現する。いずれの場合も、結果を量子化パラメタおよび量子化された標本として、ビット流形成に伝達する。

ビット流形成では、分割帯域の各々の信号に対してのハフマン符号を出力する。ここまでの結果をビットストリームに出力することで、MP3 の１フレームが出来上がる。これを繰り返すことにより、一曲分の MP3 が出来上がる。

また、符号器の入力側に遮断周波数２～１０kHz の高域通過フィルタを含んでいることが望ましい。これによって、もっとも低域側の分割帯域に対して、必要以上に多くビットを割り当てることを防ぎ、結果的に全体の音質を向上させることになる。と、仕様書には書かれている。

６、エンコードの速度の向上

MP3 がエンコードする中で、一番時間がかかる場所が、ハフマン符号器である。なぜならば、32 個もあるハフマン符号表から一つだ

け適切な符号表を選ばなければならないからである。この部分にはかなり複雑な計算が必要であり、計算ステップ数が多くなることからコンピュータで計算するにも、その分だけ時間がかかってしまう。このハフマン符号表を選び、ハフマン符号するハフマン符号器のアルゴリズムを改良することにより、エンコード速度を向上させることが出来ると考えられる。しかし、残念ながらまだ実際に動くエンコーダを作っていないため、実証することができない。

7、MP3 の今後

インターネットの普及と技術向上によって、コンピュータの技術も向上した昨今、音楽と言うものは昔みたいにテレビやラジオで聞くよりも、パソコンやポータブル機器で聞くほうが主流になりつつある。いままで CD ショップへ行かなければ、手に入れることの出来なかった、人気アーティストの新盤が、今ではインターネットを使い自宅のパソコンでボタン一つ購入できる。しかも、その場で、電線を通って手元にやってくる。そんな時代になった今、MP3 の技術は大きく社会に貢献している。MP3 がなければ、このようなミュージックライフはやってこなかっただろう。今では、MP3 のほかに、Vorbis の OGG、i-Pod などでよく使われる ATRAC、５．１チャネルのオーディオデータが作成できる AAC、Microsoft が作った WMA など、MP3 以外の圧縮オーディオ規格は沢山出てきている。最近では、MP3 がやっている人間には聞こえない高音域を削ると言うことをやめて、可逆性圧縮（情報の劣化が無い圧縮）が出来る規格まで存在する。一時期、景気がよかった頃は、大手のオーディオメーカーから、小さな町工場まで、ポータブルオーディオの新商品をこぞって出していたのに、今ではそのオーディオブームも下火になり、不況の風が追い討ちを掛けるように、この業界に襲い掛かっているようだ。しかし、圧縮音源はポータブルオーディオだけが活躍する舞台ではない。パソコンで視聴できる動画の音源部分に使われているのだ。「youtube」や「ニコニコ動画」、「veoh」などの動画サイトでも、この圧縮音源は使われている。

MP3 に限らず、圧縮音源はこれからもきっと、進化し続けるだろう。現に MP3PRO という新しい規格が誕生しようとしている。この MP3PRO は MP3 を可逆性圧縮に改良したものになりそうである。可逆性圧縮にしたほうが、従来の方式より音質面において有利ではある。しかし、ファイルサイズは少し大きくなる。

また、音楽の再生の面だけでなく、録音の面でも MP3 は非常に役に立っている。今では、小型の IC レコーダが、MP3 に直接録音できるくらいまで、技術が向上している。

8、感想

研究を本格的に始めたのが昨年の 10 月でした。非常に遅いスタートでした。ここまで来るのに、こんなにも時間がかかるとは思わず、非常に後悔しております。MP3 に関する資料が少なく、量子化などで使われる数式が難しく、それを解読するのにも時間がかかり・・・。もっと早く取り組むべきでした。すみませんでした。

９、参考文献

[1]、MP3 の復元
http://www.ctrl.nara-k.ac.jp/~idlab/2004member/koba/**mp3**.doc

[2]、日本工業標準調査会：データベース JIS 詳細表示- JISX4323
http://www.jisc.go.jp/app/pager?id=31259&%23jps.JPSH0090D:JPSO0023:/JPS/JPSO0090.jsp=&AKKNB_vJISJISNO=X4323