TTC標準
Standard

JT-G722.1

フレーム消失の少ないシステムにおける ハンズフリー用途向け 24 および 32kbit/s 低演算量符号化方式

Low-complexity coding at 24 and 32 kbit/s
for hands free operation in systems with low frame loss

第 4.1 版

2010 年 5 月 26 日制定

社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

Telecommunication Technology Committee

目　次

＜参考＞

１．　英文記述の適用レベル

適用レベル：Ｅ３

本標準の本文および図表に英文記述を含んでいる。

２．　国際勧告等との関連

本標準は、１９９９年９月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１に準拠したものである。

本標準の付属資料Ａは、２０００年２月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１ Annex Ａに準拠したものである。　本標準の付属資料Ｂは、２０００年１１月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１ Annex Ｂに準拠したものである。

また、本標準は、２００２年１０月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１に対するインプリメンターズガイドに準拠して改定されたものである。

また、本標準の付属資料Ａおよび付属資料Ｃは、２００５年４月にＩＴＵ－Ｔ　ＳＧ１６でＡＡＰに進むことが合意され、２００５年５月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１ Annex A（改定版）および Annex C に準拠したものである。

また、本標準は、２００８年６月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１に対する Corrigendum1 に準拠して改定されたものである。

３．上記国際勧告等に対する追加項目等

３．１　オプション選択項目

な　し

３．２　ナショナルマター決定項目

な　し

３．３　その他

(1)本標準は、上記ＩＴＵ－Ｔ勧告に対し、先行している項目はない。

(2)本標準は、上記ＩＴＵ－Ｔ勧告に対し、追加した項目はない。

(3)本標準は、上記ＩＴＵ－Ｔ勧告に対し、削除した項目はない。

(4)本標準は、上記ＩＴＵ－Ｔ勧告に対し、変更した項目はない。

３．４　原勧告との章立て構成比較

上記国際勧告等との章立て構成の相違はない。

４．改版の履歴

版　数	制　定　日	改　版　内　容
第 1 版	2000 年 4 月 20 日	制定
第２版	2000 年 11 月 30 日	付属資料Ａの追加
第 3 版	2001 年　4 月　19 日	付属資料Ｂの追加
第 3.1 版	2003 年　11 月　27 日	C コード改定に関する記述の追加
第４版	2005 年　11 月 24 日	付属資料Ａの改定および付属資料 C の追加
第 4.1 版	2010 年　5 月　26 日	C コード改定に関する記述の追加

５．工業所有権

本標準に関わる「工業所有権の実施の権利に係る確認書」の提出状況は、ＴＴＣホームページでご覧になれます。

６．その他

(l) 参照している勧告、標準等

ＴＴＣ標準：　　ＪＴ－Ｇ７２２、ＪＴ－Ｈ３２３、ＪＴ－Ｈ２４５、ＪＴ－Ｈ２２５．０

ＩＴＵ－Ｔ勧告：　　ＩＴＵ－Ｔ勧告Ｇ．１９２

ＩＳＯ／ＩＥＣ標準：ＩＳＯ／ＩＥＣ９８９９

(2) ＴＴＣ標準ＪＴ－Ｇ７２２．１は、ＩＴＵ－Ｔ勧告Ｇ．７２２．１に準拠しており、本標準中で言及しているＣコードとは、ＩＴＵ－Ｔ勧告Ｇ．７２２．１のものをさし、ＩＴＵ－ＴのWebサイトから入手可能である。

本標準本体および付属資料Bにて参照しているCコードの改定に関して、２００２年１０月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１に対するインプリメンターズガイドに記述がある。

また、付属資料Bにて参照しているCコードの改定に関して、２００８年６月に承認されたＩＴＵ－Ｔ勧告Ｇ．７２２．１に対するCorrigendum1に記述がある。

１． 本標準の規定範囲

本標準は、7kHz 帯域幅のオーディオ信号に対して 24kbit/s または 32kbit/s で動作する低演算量の符号器および復号器に関して記述されたものである。更に、本アルゴリズムは、フレーム消失の発生確率が低い条件において、会議システムのようなハンズフリーのアプリケーションに使用されることが推奨される。音声または音楽の入力に対して使用される。ビットレートは、任意の 20ms フレーム単位で切り替えることができる。

２．概要

本標準は、24kbit/s または 32kbit/s のビットレートで動作し、50Hz から 7kHz のオーディオ帯域の信号を提供するディジタル広帯域符号化アルゴリズムを記述したものである。符号器へのディジタル入力は、サンプリング周波数が 16kHz の 14,15 または 16 ビットの２の補数形式である（ＴＴＣ標準ＪＴ－Ｇ７２２と同様）。符号器入力および復号器出力におけるアナログとディジタルのインタフェース回路は、ＴＴＣ標準ＪＴ－Ｇ７２２の記載と同一の仕様に従うべきである。

アルゴリズムは、変調重複変換（ＭＬＴ）を用いた変換技術に基づいたものであり、20ms フレーム（320 サンプル）で動作する。変換窓長（基底関数長）が 640 サンプルで、フレーム間の重複が 50%（320 サンプル）であるので、有効な先読みのバッファサイズは 20ms である。従って、合計のアルゴリズム遅延は、フレームサイズと先読みの合計で 40ms である。他の全ての遅延は、処理遅延と網伝送遅延によるものである。

本標準の符号化アルゴリズムは、ビットイグザクトな固定小数点演算で記述されている。第 6 章にて示されるＣコードは、本標準の一部をなしており、ビットイグザクトな固定小数点演算を実現したものとして、ＩＴＵ－Ｔの Web サイトから入手可能である。なお、第４章および第５章の数学的な記述との相違点がある場合には、Ｃコードの方が優先される。

符号器（第４章）および復号器（第５章）の数学的な記述は、他の何通りかの方法で実現することは可能である。しかし、第６章のＣコードは参照用に提供されたものであり、本標準に準拠するためには、いかなる実現方法によっても、任意の入力信号が第６章のＣコードと同一の出力結果を出力するものでなければならない。

上記目的を達成するためには、本標準の実現に際しては、第６章のＣコードにより与えられる計算の詳細、定数テーブル、変数の適応化の順序に従うべきである。本アルゴリズムには、正しいビットイグザクト動作を維持するためにきわめて重要な箇所が多く存在する。そのような箇所は、第６章のＣコードにより記述される計算の詳細、定数テーブル、変数の適応化の順序と完全に同一に実現しなければならない。

Ｃコードは参照用に提供されたものであり、特定の実現対象のプラットフォーム用として（メモリや演算量等の面で）最適化されたものではない。Ｃコードは、特定の実現に対しては最適化が必要である。

非網羅的なテスト信号のセットが、本標準の一部として、符号器および復号器の実現が本標準に適合していることを検証するためのツールとして提供される。

実際面において、購入者は、装置またはソフトウェアが相互接続性を保証する本標準に準拠していることを期待する。本標準を実現する際には、Ｃコードの最適化、もしくはＣコードの修正を選択することもできる。その場合には、任意の入力信号に対して、第６章のＣコードで期待される出力と同一結果を出力することを検証すべきである。

３．参考とする標準

下記のＴＴＣ標準および他の参考文献は、本標準での参照を通して本標準の規定を構成するものである。本標準の出版の際には、示された版が有効である。全ての標準および参考文献は、改定に従うものとする。従って、本標準のユーザには、最新の版が適用されるよう奨励される。

(1) ＴＴＣ標準ＪＴ－Ｇ７２２（1994）
64kbit/s 以下の 7kHz オーディオ符号化方式

(2) ＩＴＵ－Ｔ勧告Ｇ．１９２（03/96）

A Common digital parallel interface for speech standardization activities

(3) ISO/IEC 9899:1990

Programming Languages – C

４． 符号器

Figure 1／JT-G722.1 に、符号器のブロック図を示す。

20ms（320 サンプル）毎に、最新の時間領域の 640 オーディオサンプルが変調重複変換（ＭＬＴ）に供給される。各々の変換により、320 個のＭＬＴ係数からなるフレームが生成され、そのＭＬＴ係数フレームはそれぞれ独立に符号化される。すなわち、前フレームに依存する状態情報は存在しない。24kbit/s および 32kbit/s 動作に対して、フレーム当たりの割り当てビット数はそれぞれ、480 および 640 である。

ＭＬＴ変換によって生成された変換係数は、最初に、振幅包絡を計算して量子化するモジュールに適用される（Figure２／JT-G722.1 参照）。振幅包絡は、ＭＬＴスペクトルの粗い表現である。スペクトルは領域と呼ばれる 20 個のＭＬＴ係数からなるブロックに分割される。各々の領域は 500Hz 帯域幅を表す。全帯域幅が 7kHz なので、*number_of_regions* は 14 に設定される。7kHz を超える周波数を表すＭＬＴ係数は無視される。振幅包絡符号化ビットは、ＭＵＸ（多重化器）へ送られ復号器に送信される。振幅包絡の量子化および符号化後に残ったビットは、カテゴリ化手順においてＭＬＴ係数を符号化するために使用される。

量子化された振幅包絡および振幅包絡の符号化後におけるフレームの残り（４ビットのカテゴリ化制御ビットを除く）のビット数を用いて、カテゴリ化手順に従い 16 セットのカテゴリ化タイプ（カテゴリ化タイプ０からカテゴリ化タイプ 15）を生成する。各々のカテゴリ化タイプにおいて、同一のＭＬＴ係数を符号化するのに各々の異なるビット数を必要とする。

各々のカテゴリ化タイプは、14 のカテゴリ割り当てのセットからなり、14 の領域の各々に１つのカテゴリが割り当てられる。カテゴリは領域に対して事前に決定された量子化および符号化のパラメータのセットを定義する。領域を符号化するのに必要な所望ビット数は、各々のカテゴリに関連している。本符号器は可変長のハフマン符号化を用いるため、最終的な使用ビット数は領域におけるＭＬＴ係数の個々の系列に依存して変わる。

次に、ＭＬＴ係数に対して、16 の計算されたカテゴリ化タイプの各々において、異なる量子化および符号化が行われる。各々のカテゴリ化タイプに対して、実際に必要な符号化ビット数が決定される。

量子化および符号化は領域毎に行われる。カテゴリ化は、14 の全ての領域に対してカテゴリ割り当てを決定し、振幅包絡とともに各々の領域に対するカテゴリ割り当ては、その領域における 20 の全てのＭＬＴ係数に使用される全ての量子化および符号化のパラメータを決定する。

領域におけるＭＬＴ係数は、最初に、その領域における量子化された振幅包絡により正規化され、それからスカラ量子化される。その結果得られたスカラ量子化インデックスはベクトルインデックスとして構成される。ベクトルインデックスはその後ハフマン符号化される。従って、それらは可変ビット数で符号化される。最も頻度の高いインデックスは頻度の低いベクトルインデックスに比べてより少ないビットを必要とする。

本コーデックは可変長のハフマン符号化を用い、固定の伝送ビットレートが要求されるため、ビットレートを伝送路のレートに拘束させる方法が必要である。４ビットのカテゴリ化制御ビットが、復号器に対してどのカテゴリ化タイプが選択されたかを特定する。カテゴリ化切り替えが、送信のためＭＵＸに出力する符号化ビット（選択されたカテゴリ化タイプを用いて生成された量子化ＭＬＴ係数を表す）を指定する。伝送路のレートに最も近いビット数を供給するカテゴリ化タイプが送信のために選択される。

４．１ 変調重複変換（ＭＬＴ）

ＭＬＴは完全再生する線形変換であり、隣接するＭＬＴフレームの基底関数はお互いに 50%の重複をしている。ＭＬＴへ入力するオーディオ信号は最新の 640 個のオーディオサンプルであり、$x(n)$と定義する。

但し

$x(0)$は最も古いオーディオサンプル

また

$0 \le n < 640$

このとき、ＭＬＴ係数は 320 個の係数を持ち $mlt(m)$と定義する。

但し

$0 \le m < 320$

ＭＬＴ係数は以下によって与えられる。

$$mlt(m) = \sum_{n=0}^{639} \sqrt{\frac{2}{320}} \sin\left(\frac{\pi}{640}(n+0.5)\right) \cos\left(\frac{\pi}{320}(n-159.5)(m+0.5)\right) x(n)$$

ＭＬＴは窓かけおよび重ねあわせ加算演算を行い、タイプⅣ 離散コサイン変換（ＤＣＴ）を行う。その窓かけおよび重ねあわせ加算演算は以下となる。

$$v(n) = w(159-n)x(159-n) + w(160+n)x(160+n)$$

但し

$0 \le n \le 159$

$$v(n+160) = w(319-n)x(320+n) - w(n)x(639-n)$$

但し

$0 \le n \le 159$

ここで

$$w(n) = \sin\left(\frac{\pi}{640}(n+0.5)\right)$$

但し

$0 \le n < 320$

$v(n)$をタイプⅣ ＤＣＴと組み合わせることで、結果的にＭＬＴに対する式は以下となる。

$$mlt(n) = \sum_{n=0}^{319} \sqrt{\frac{2}{320}} \cos\left(\frac{\pi}{320}(n+0.5)(m+0.5)\right) v(n)$$

但し、ＤＣＴの演算量を著しく減らすために、高速変換技術を使用する。

４．２　振幅包絡の計算と量子化

ＭＬＴ係数は 20 個の係数毎に領域 r として区分される。また、総領域は $number\_of\_regions$=14 である。よって、領域 r はＭＬＴ係数 $20r$ から $20r$+19 を含む。

ここで

$0 \le r < number\_of\_regions$

但し、7kHz を超える周波数は対象とする帯域外であるので、最高周波数から 40 個のＭＬＴ係数は使用しな

い。領域 *r* における振幅包絡はその領域内のＭＬＴ係数の実効値として定義され、以下のように計算される。

$$rms(r) = \sqrt{\frac{1}{20}\sum_{n=0}^{19} mlt(20r+n)mlt(20r+n)}$$

ここで、*rms*(*r*)の量子化を行う。量子化器の出力インデックスを *rms_index*(*r*)とする。また、*rms*(*r*)の量子化に伴い、再生される量子化値は下記のようになる。

$2^{\left(\frac{i+2}{2}\right)}$

但し

i は整数値

ここで

$-8 \le i \le 31$

さらに、*rms_index*(0)は下記のように制限される。

$1 \le rms\_index(0) \le 31$

rms(*r*)の量子化は対数スケールで行い、$2^{\left(\frac{i+2}{2}\right)}$ となる値が求められたとき、量子化の範囲は $2^{\left(\frac{i-0.5+2}{2}\right)}$ から $2^{\left(\frac{i+0.5+2}{2}\right)}$ となる。

例えば、*rms(r)*=310 のとき、量子化レベルは $2^{\left(\frac{15+2}{2}\right)}$、即ち、362.04 となる。また、*rms_index* (*r)*=15 となる。その理由として $2^{\left(\frac{15-0.5+2}{2}\right)}$=304.43 となるためである。

４．３　振幅包絡の符号化

rms_index(0)はそれぞれのフレームで送信される最初の値であり、５ビットが割り当てられる。また、最上位ビットを最初に送信する。*rms_index*(0)=0 という値は予約されていて使うことはできない。残りの 13 個の振幅包絡の出力インデックスは送信するために差分符号化およびハフマン符号化される。振幅包絡が符号化されるときの隣り合うインデックスの最大許容差は+11 から−12 である。この範囲内に差を収めるために、谷の部分は頂点が正確に表現されるように上向きに調節される。このことは以下の擬似Ｃコードに記されている。

```
for (r=number_of_regions−2; r>=0; r−−)
{
    if (rms_index[r]<rms_index[r+1]−11)
        rms_index[r]=rms_index[r+1]−11;
}
for (r=1; r<number_of_regions; r++)
{
    j=rms_index[r]−rms_index[r−1];
    if (j<−12)
    {
        j=−12;
        rms_index[r]=rms_index[r−1]+j;
```

```
        }
        differential_rms_index[r]= j;
    }
```

実効値インデックスの差 *differential_rms_index*[*r*]は領域 *r* の順番に送信される。*differential_rms_index*[*r*]はテーブル *differential_region_power_codes*[*r*][*j*+12]で定義される可変長ハフマン符号に従い符号化される。また、ハフマン符号化のビット数はテーブル *differential_region_power_bits*[*r*][*j*+12]により求められる。この配列は本標準のＣコードに含まれている。それぞれの領域は特定のハフマン符号に関連付けられる。ハフマン符号の送信は最上位ビットより行う。

４．４　カテゴリ化手順

カテゴリ化手順はＭＬＴ係数を量子化するために使われるステップサイズ（と他の量子化と符号化に関連するパラメータ）を決定する。カテゴリ化の手順は最初にそれぞれの領域毎にカテゴリを割り当てる。0－7 までの８つのカテゴリがある。16 の異なるカテゴリ化タイプの組が計算され、最後に送信のために１つに確定される。このカテゴリ化手順は復号化でも使用する。また、同じ入力が与えられた場合、この手順を実現した種々の機器で同一のカテゴリ化タイプとならなければならない。このことは相互接続性のために重要である。この手順における入力は以下のようになる。

(1)　*number_of_available_bits*:振幅包絡とカテゴリ化制御ビットを計算したあとでもまだ使われていないフレーム内の実際のビット数

(2)　*rms_index*()：ＭＬＴ係数の実効値 *rms*(*r*) の全ての領域で量子化された値

領域に割り当てられたカテゴリは領域の量子化と符号化のパラメータを決定する。また、それぞれの領域に割り当てられたカテゴリは領域の量子化されたＭＬＴ係数を表わすのに必要な総ビット数をも決定する。これは、可変長ハフマン符号が使われるので、実際のビット数は領域のＭＬＴ係数の統計値により変わる。よって、16 のカテゴリ化候補の中から、後に述べる基準にしたがって、最適なカテゴリ化タイプを送信のために選択する。それぞれのカテゴリ(0-7)の予測されるビット数を Table４－１／JT-G722.1 に定義する。

Table４－１／JT-G722.1　Expected number of bits for each category (ITU-T G.722.1)

category	code bits per region as a function of category (refer to *expected_bits_table*[] in the C code)
0	52
1	47
2	43
3	37
4	29
5	22
6	16
7	0

４．４．１　使用可能なビット数の調整

実際の使用可能なビット数に基づいて、使用可能なビット数の推定値を以下に算出する。

```
if
```

number_of_available_bits > 320

then

estimated_number_of_available_bits = 320+((*number_of_available_bits* –320) * 5/8)

estimated_number_of_available_bits は、カテゴリ化の過程において、常に実際の使用可能なビット数未満であり余裕を持たせている。

４．４．２　最初のカテゴリ化タイプの算出

-32～31 の任意の整数 *offset* に対して、カテゴリの割り当てを以下に示す。

$category(r) = \mathrm{MAX}\{0,\mathrm{MIN}\{7,(offset\text{-}rms\_index(r))/2\}\}$

ここで $0 \le r < number\_of\_regions$

全領域に対して、同一の *offset* を使用する。予測されるＭＬＴ符号語ビットの総数を次式に示す。

$$expected\_number\_of\_code\_bits = \sum_{r=0}^{13} expected\_bits\_table(category(r))$$

次式を満足する最大 *offset* が見つけられるまで *offset* の値を調整する。

$expected\_number\_of\_code\_bits \ge estimated\_number\_of\_available\_bits - 32$

４．４．３　残りの 15 通りのカテゴリ化タイプの生成

一度最初のカテゴリ化タイプを算出すると、残りの 15 通りのカテゴリ化タイプが算出される。それぞれの新しいカテゴリ化タイプについては、直前のカテゴリ化タイプと関連する 1 つの領域だけについてカテゴリを調整する。残りのカテゴリ化タイプを決定する方法を以下に示す。

$initial\_categorization(r)=\mathrm{MAX}\{0,\mathrm{MIN}\{7,(offset\text{-}rms\_index(r))/2\}\}$

ここで $0 \le r < number\_of\_regions$

一時的な変数を以下に定義する。

max_category(*r*)

max_bits

min_category(*r*)

min_bits

max_category(*r*)=*initial_categorization*(*r*)

min_category(*r*)=*initial_categorization*(*r*)

max_bits = *expected_number_of_code_bits*

min_bits = *expected_number_of_code_bits*

残りの 15 通りのカテゴリ化タイプ毎に以下の比較を行う。

if

$max\_bits + min\_bits \le 2*estimated\_number\_of\_available\_bits$

then

新しいカテゴリ化タイプはより大きな予測ビット数が必要である。以下の式を満足する領域 *r* について

max_category(*r*)>0

次式を最小にする領域を探索する。

offset-rms_index(*r*)-2**max_category*(*r*)

上式を最小にする領域が複数存在する場合、最小の領域 *r*（最低周波数）を選択する。

その領域のカテゴリ値 *max_category*(*r*)を１だけ減少する。新しいカテゴリ化タイプの予測ビット数を再算出して *max_bits* に設定する。

Otherwise

新しいカテゴリ化タイプはより小さな予測ビット数が必要である。以下の式を満足する領域 *r* について

min_category(*r*)<7

次式を最大にする領域を探索する。

offset-rms_index(*r*)-2**min_category*(*r*)

上式を最大にする領域が複数存在する場合、最大の領域 *r*（最大周波数）を選択する。

その領域のカテゴリ値 *min_category*(*r*)を１だけ増加する。新しいカテゴリ化タイプの予測ビット数を再算出して *min_bits* に設定する。

このように 16 通りの一意のカテゴリ化タイプが生成される。第７章に詳述されるように、カテゴリ化タイプは予測ビット数による順番となる。カテゴリ化タイプ０は、最大予測ビット数を保有し、カテゴリ化タイプ 15 は最小予測ビット数を保有する。それぞれのカテゴリ化タイプは、カテゴリ値が１だけ異なる１つの領域を除いて、前後のカテゴリ化タイプと同様である。例えば、カテゴリ化タイプ７において領域５はカテゴリ２が設定され、カテゴリ化タイプ８において領域５はカテゴリ３に設定され、他の領域についてはカテゴリ化タイプ７と同様である。

カテゴリ化手順の詳細フローチャートを第７章に示す。

４．５　スカラ量子化ベクトルハフマン符号化（ＳＱＶＨ）

カテゴリ値 0～6 に割り当てられた領域について、ＭＬＴ係数を符号部と振幅部に分割する。振幅部を *rms*(*r*) の量子化値で正規化する。次に下式で示されるスカラ量子化を行いベクトルに結合しハフマン符号化する。カテゴリ７に割り当てられた領域は、このように処理されずに１ビットも送信されない。

符号器は、領域 *r* 毎にＭＬＴ係数 *mlt*(*i*)の絶対値の正規化、量子化を行い量子化インデックス *k*(*i*)を生成する。

k(*i*)=MIN{whole number of part (*x** absolute value of (*mlt*(20*r*+*i*))+*deadzone_rounding*),*kmax*}

ここで特定の領域におけるインデックスは以下の範囲である。

$0 \le i < 20$

また

x= 1/ (*stepsize* * (quantized value of *rms*(*r*))

また

stepsize、*deadzone_rounding*、および、*kmax* を Table４－２／JT-G722.1 に定義

Table４－２／JT-G722.1　Table of constants used by the SQVH procedure (ITU-T G.722.1)

category	stepsize	deadzone_rounding	kmax
0	$2^{-1.5}$	0.3	13

1	$2^{-1.0}$	0.33	9
2	$2^{-.5}$	0.36	6
3	$2^{0.0}$	0.39	4
4	$2^{.5}$	0.42	3
5	$2^{1.0}$	0.45	2
6	$2^{1.5}$	0.5	1

インデックス *k*()をベクトルインデックスに結合する。ベクトルの特性はカテゴリごとに異なる。Table４－３／JT-G722.1 および Figure３／JT-G722.1 に示すように、領域毎に *vpr* 個の *vd* 次元の定義済みベクトルが存在する。スカラ値 *k*()は、以下に示すインデックスにより特定された一意のベクトルに対応する。

$$vector\_index(n) = \sum_{j=0}^{vd-1} k(n \times vd + j)(kmax+1)^{(vd-(j+1))}$$

ここで

$0 \le n \le vpr-1$ は領域 *r* の *n* 番目ベクトル

また

j=領域 *r* におけるベクトルの *k*()の *j* 番目の値に対するインデックス

vd=カテゴリのベクトル次元

vpr=カテゴリにおける領域当たりのベクトル数

kmax=Table４－３／JT-G722.1 に示すカテゴリに対する *k*()の最大値

Table４－３／JT-G722.1 に、*vd*、*vpr* および *u* の値を示す。ここで、$u=(kmax+1)^{vd}$ は、あるカテゴリにおけるベクトルのとり得る値の数を示す。

Table４－３／JT-G722.1 Definition of constants *vd*,*vpr* and *u*

(ITU-T G.722.1)

category	*vd*	*vpr*	*u*
0	2	10	196
1	2	10	100
2	2	10	49
3	4	5	625
4	4	5	256
5	5	4	243
6	5	4	32

あるカテゴリについて、ベクトル *vector_index*(*n*)を表すために必要とするビット数を、テーブル *mlt_sqvh_bitcount_category_0*[] ～ *mlt_sqvh_bitcount_category_6*[] に与える。このテーブルは、*mlt_svqh_code_category_0*[]～*mlt_svqh_code_category_6*[]において、対応する符号語が必要とするビット数を与える。このビット数には符号ビットは含まれない。*k*()=0 の値は符号ビットを必要としない。

カテゴリ *y* の領域 *r* に対するＭＬＴ係数を表すために、実際に必要とするビット数（符号ビットを含む）を以下に示す。

$$number\_of\_region\_bits(r) = \sum_{n=0}^{vpr-1} mlt\_svqh\_bitcount\_category\_y(vector\_index(n)) + (\text{number of sign bits in } n^{th} \text{ vector})$$

４．６　レート制御

フレームを表すために実際に必要とされるビット総数は、各カテゴリ化タイプごとに算出される。これは、振幅包絡を表すために使われるビットと４ビットのカテゴリ化制御ビットとＭＬＴ係数を表すのに必要となるビットを含んでいる。そして、送信に最適なカテゴリ化タイプを選択し、カテゴリ化制御ビットを用いてこの選択を示す。

最初に、割り当て超過のビット総数を持つカテゴリ化タイプが除外される。残りのカテゴリ化タイプの中で最小のインデックスを持つものが選択される。例えば、０から３までのカテゴリ化タイプが割り当てを超えたビットを使用し、カテゴリ化タイプ４は割り当てられたビットに収まっている時、カテゴリ化タイプ４が選択される。

もし、どのカテゴリ化タイプも割り当てられたビットに収まらないならば、最も近いカテゴリ化タイプ（通常は15）が選択される。それから、フレームに割り当てられたビットが尽きるまで符号語ビットが送信される。

オーディオの20msのフレームを表すために符号器が必要とするビット数が、１フレームあたりに割り当てられるビット数（480か640ビット）より少ない場合が起こりうる。この場合、ビット列の最後にある使用されない残りのビットは、全て１に設定される。

４．７　ＭＬＴベクトルインデックスの送信

ベクトルインデックスは、低い周波数から高い周波数に向かって周波数順に送信される。それらは、Ｃコード中の配列 *mlt_svqh_bitcount_category_x*[] と *mlt_svqh_code_category_x*[] (*_x* は、$0 \le x \le 6$ のカテゴリ値を表している)によって定義される可変長符号により符号化される。最も左側の（すなわち最上位）ビットが最初に送信される。それぞれのベクトルのゼロでないＭＬＴ係数に関連した符号ビットは、各ベクトルインデックスの可変長符号の送信直後に送信される。符号ビットもまた、周波数順に送信される。符号ビットは、正の数に対して１に設定される。

ビット列

フレーム中のビット総数は、480 か 640 であり、それぞれ 24kbit/s と 32kbit/s に対応している。フレーム中のビット総数は固定されているが、カテゴリ化制御ビットパラメータを除いて他の全てのパラメータは可変長符号すなわち可変なビット数で表される。Figure４／JT-G722.1 は、この点と、送信されるパラメータフィールドの順番を図示している。全ての可変長符号とカテゴリ化制御ビットは、左側（最上位）のビットから右側（最下位）のビットの順に送信される。

５．　復号器

全てのフレームに対して初めに、領域０の振幅インデックスを表している最初の５ビットが復号される。それから残りの領域が、ハフマン復号され再構成される。４つのカテゴリ化制御ビットは、16 の可能なカテゴリ化タイプのうちどれが符号器によって選択され送信されたかを決定するために復号される。フレーム中の残りの符号化ビットは、量子化されたＭＬＴ係数を表しており、各領域のカテゴリ情報に従って復号される。符号器におけるのと全く同様に、復号器におけるカテゴリ化手順は、復号されずに残っているビット数（現フレーム中の）とともに振幅包絡を使用し、16 の可能なカテゴリ化タイプの集合を算出する。

いくつかの領域に対して、符号器によりカテゴリ７が割り当てられる場合がある。これは、これらの領域を表すためにはＭＬＴ係数が、全く送信されないということを意味している。カテゴリ７を割り当てられた領域は雑音充填と呼ばれる技術を用いて再構成される。これらの領域に対する平均的なＭＬＴ係数の大きさは、振幅包絡から得ることができる。カテゴリ７のＭＬＴ係数を０に設定する代わりに、復号器はその領域に対する

平均的なＭＬＴ係数の大きさに比例した値に設定する。各係数の符号は、ランダムに設定される。係数の符号の決定は、何らかの方法によってなされる。例えば、単純な擬似乱数発生器で、十分である。

カテゴリ５と６では、ＭＬＴ係数の多くがゼロに量子化される場合があるため、雑音充填は、これらのカテゴリにも適用される。ゼロとして送信された値は、その領域に対する平均振幅の何分の一かに設定される。符号は再度ランダムに決定される。

ゼロ以外の値にスカラ量子化された係数に対して、正規化された係数は、前もって定義されているテーブルから再構成される。再構成された値は、それから適切な *rms*(*r*)の値を使用して縮尺される。7kHz 以上の周波数を表している 40 個のＭＬＴ係数はゼロに設定される。ＭＬＴ係数の再構成の後、逆変調重複変換（ＩＭＬＴ）は、320 個の新しい時間領域のサンプルを生成する。

ＩＭＬＴの最後の重ね合わせ加算演算を除いて、復号器によって受け取られる各フレーム中の情報は、前フレーム中の情報には依存しない。

５．１　振幅包絡の復号

フレームの最初の５ビットは、*rms_index*(*0*)を表している。それから、残った領域に対して、*differential_rms_index*(*r*) に対する可変長符号が、Ｃコード中で参照されている配列 *differential_region_power_bits*[][]と配列 *differential_region_power_codes*[][]に従って復号される。これらの領域に対する量子化器インデックスは、次のように再構成される。

$$rms\_index(r) = rms\_index(r-1) + differntial\_rms\_index(r)$$

ここで、

$$1 \le r < number\_of\_regions$$

カテゴリ化タイプの決定

振幅包絡を復号した後、復号器はＭＬＴ係数を表すために残っているビットの数を決定する。これは、次のようになされる。

bits available = bits per frame – amplitude envelope bits – four (categorization control bits)

符号器と同じカテゴリ化手順を用いて、16 個の可能なカテゴリ化タイプが算出される。４ビットのカテゴリ化制御ビットは、どのカテゴリ化タイプがＭＬＴ係数を符号化するのに使用されたかを示しており、復号器によっても、使用されるべきである。

ＭＬＴ係数の復号

各領域において、ＭＬＴベクトルに対応する可変長符号は、適切なカテゴリテーブルに従い復号される。Ｃコード中の配列 *mlt_svqh_bitcount_category_x*[]と *mlt_svqh_code_category_x*[]は、この目的のために使用される（ここで *_x* は $0 \le x \le 6$ のカテゴリ値を表す）。領域内の個々のＭＬＴ係数量子化インデックス *k*(*i*)は、ベクトルインデックスから次のように再生される。

$$k(i) = \left\lfloor \frac{vector\_index(n)}{(kmax+1)^j} \right\rfloor \mathrm{MOD}(kmax+1)$$

ここで

$\lfloor z \rfloor$ は、z を超えない最大の整数を示す

$i = (n+1)vd - j - 1$

$0 \le j \le vd - 1$

$0 \le n \le vpr - 1$、領域 r における n 番目のベクトルを表す

また

vd = 与えられたカテゴリのベクトル次元

kmax = Table４－２／JT-G722.1 に示す与えられたカテゴリに対する *k*()の最大値

ＭＬＴ係数の再生には、Ｃコード中の配列 *mlt_quant_centroid*[][]のセントロイドテーブルを使用する。ＭＬＴ係数の大きさは、対象領域の *rms*(*r*)と復号されたベクトルインデックスで指定されたセントロイドとの積を計算することにより再生される。正負は符号ビットに従う。

雑音充填

カテゴリ７に割り当てられた領域では、ＭＬＴ係数は符号化されない。カテゴリ５および６では、大きな量子化ステップサイズのため、大部分のＭＬＴ係数がゼロとして符号化される。これらのゼロはランダムな符号と *rms*(*r*)に比例した値の係数で置き換えられる。比例定数は Table５－１／JT-G722.1 に定義する。

Table５－１／JT-G722.1　Noise Fill proportionality constants

(ITU-T G.722.1)

category	default noise-fill proportionality constant
5	.176777
6	.25
7	.707107

５．５　不足ビット

最後の非カテゴリ７領域を符号化し終える前に、符号器がビットを使い果たすフレームがあるかもしれない。このような場合の復号器の動作は、その領域と残りすべての領域をカテゴリ７割り当てで処理する。

５．６　フレーム消失

もし復号器がフレーム消失やフレーム誤りの通知を受けたら（本標準では規定されていない外部シグナリングメカニズムによって）、復号器は前フレームの復号ＭＬＴ係数を繰り返す。この処理は、これらの係数を時間領域に変換し、前フレームと次フレームの復号された情報を重ね合わせ加算演算をすることにより行われる。もし、前フレームもフレーム消失やフレーム誤りならば、そのとき復号器はすべての現フレームのＭＬＴ係数をゼロに設定する。

５．７　逆ＭＬＴ（ＩＭＬＴ）

それぞれの逆ＭＬＴ操作は、320 個の時間領域オーディオサンプルを生成するために 320 個の係数を処理する。逆ＭＬＴは、タイプⅣのＤＣＴとそれに続く窓かけ、重ね合わせ加算演算に分解できる。

タイプⅣ ＤＣＴは、次式で表される。

$$u(n) = \sum_{m=0}^{319} \sqrt{\frac{2}{320}} \cos\left(\frac{\pi}{320}(m+0.5)(n+0.5)\right) mlt(m)$$

窓かけ、重ね合わせ加算演算は、現フレームのＤＣＴ出力の半分のサンプルと前フレームのＤＣＴ出力の半分のサンプルとを使用する。

$$y(n) = w(n)u(159-n) + w(319-n)u\_old(n)$$

但し

$$0 \le n \le 159$$

$$y(n+160) = w(160+n)u(n) - w(159-n)u\_old(159-n)$$

但し

$$0 \le n \le 159$$

ここで

$$w(n) = \sin\left(\frac{\pi}{640}(n + 0.5)\right)$$

但し

$$0 \le n \le 319$$

u()の使用されなかった半分は、次フレームで使用するために、*u_old*()として保存される。

$$u\_old(n) = u(n + 160)$$

但し

$$0 \le n \le 159$$

６．Ｃコード

付属のＣコードは、本標準の必須部分であり、いくつかのファイルに分割されている。これらのファイルを下記に示す。

basic_op.c
coef2sam.c
common.c
count.c
dct4_a.c
dct4_s.c
decode.c
decoder.c
dct4_a.h
encode.c
encoder.c
huff_tab.c
sam2coef.c
tables.c

basic_op.h
count.h
dct4_s.h
defs.h
huff_def.h
huff_tab.h
tables.h
typedefs.h

それぞれのプログラムは、コンパイルされて符号器ファイル encode と復号器ファイル decode になり、これらは次のようにコマンドラインで使えるようになる。

encode　bit-stream-type　input-audio-file　output-bit-stream-file　bit-rate

decode　bit-stream-type　input-bit-stream-file　output-audio-file　bit-rate

ここで

bit-stream-type　= 0, 圧縮されたビット列を指定

bit-stream-type　= 1, テスト用のＩＴＵ－Ｔ勧告Ｇ．１９２のビット列フォーマットを指定

input-audio-file = サンプルデータとして読み込ませる 16 ビットＰＣＭオーディオファイル名

output-audio-file = 復号出力を保存する 16 ビットＰＣＭオーディオファイル名

input-bit-stream-file = 入力ビット列として読み込ませるファイル名

output-bit-stream-file = 符号化ビット列出力を保存するファイル名

bit-rate = 24000 あるいは 32000, それぞれ 24kbit/s、32kbit/s 動作に対応

７．カテゴリ化手順のフローチャート

本章では、符号器および復号器で使用されるカテゴリ化計算の詳細手順について述べる。この手順は、フローチャートに示される様に、25 のステップに分割されている。

本章では、次の変数が定義、参照されている。

$category.X[r]$

は、カテゴリ化タイプ X で領域 r に割り当てられたカテゴリである。

ここで、

$0 \le X < 16$

$0 \le r < 14 (number\_of\_regions)$

$0 \le category.X[r] \le 7$

STEP(0)

フレームあたりに割り当てるビット数（例えば、24kbit/s 動作では 480）から始めて、使用可能なビット数を計算する。そして、振幅包絡を表すためのビット数を差し引き、カテゴリ化タイプ選択を表すためのカテゴリ化制御ビットを差し引く。この数値を次のように修正する。

if

number_of_available_bits > 320

then

estimated_number_of_available_bits = 320+((*number_of_available_bits* - 320) * 5/8)

（これは、異なるビットレートでのカテゴリ割り当てに関連した統計上の差異を補償する。）

STEP(1)

一時的な配列の割り当て

initial_categorization[*number_of_regions*]

max_category[*number_of_regions*]

min_category[*number_of_regions*]

temp_category_balances[32]

一時的な変数の割り当て

offset

delta

expected_bits

max_expected_bits

min_expected_bits

max_pointer

min_pointer

categorization_count

STEP(2)

初期化

offset = -32

delta = 32

STEP(3)

それぞれの領域に対して制限を持たないカテゴリ割り当てを計算

initial_categorization[*r*] = (*offset* + *delta* - *rms_index*[*r*]) / 2

STEP(4)

それぞれの領域に対して *initial_categorization*[]を制限

if

initial_categorization[*r*] < 0

then

initial_categorization[*r*] = 0

if

initial_categorization[*r*] > 7

then

initial_categorization[*r*] = 7

STEP(5)

expected_bits_table[8]は、それぞれのカテゴリに対する平均ビット数が含まれた予め決定されたテーブルである。これは、このカテゴリ化の為の総ビット数の予測値を計算するために使用される

$$expected\_bits = \sum_{r=0}^{number\_of\_regions-1} expected\_bits\_table[initial\_categorization[r]]$$

STEP(6)

if

expected_bits ≥ *estimated_number_of_available_bits* - 32

then

offset = *offset* + *delta*

STEP(7)

delta = *delta* / 2

STEP(8)

if

delta > 0

then

go to STEP(3)

otherwise

continue on to STEP(9)

STEP(9)

initial_categorization[*r*] = (*offset* - *rms_index*[*r*]) / 2

STEP(10)

STEP(4)と同様の方法で、それぞれの領域に対して *initial_categorization*[*r*]を制限。

STEP(11)

STEP(5)と同様の方法で、*initial_categorization*[]に対する *expected_bits* を算出。

STEP(12)

初期化

max_category[*r*] = *initial_categorization*[*r*]

min_category[*r*] = *initial_categorization*[*r*]

max_bits = *expected_bits*

min_bits = *expected_bits*

max_pointer = 16

min_pointer = 16

categorization_count = 1

STEP(13)

if

max_bits + *min_bits* ≤ 2 * *estimated_number_of_available_bits*

then

go to STEP(14)

otherwise

go to STEP(16)

STEP(14)

以下の式を満たす領域 *r* に対して

max_category[*r*] > 0

次に示す関数を最小にする領域を探索する。

offset - *rms_index*[*r*] - 2 * *max_category*[*r*]

この関数を最小にする *r* が複数存在する場合は、最も小さい（すなわち、最低周波数の）領域 *r* を選択する。

STEP(15)

max_pointer = max_pointer - 1

temp_category_balances[*max_pointer*] = *r*

max_bits = *max_bits* - *expected_bits_table*[*max_category*[*r*]]

max_category[*r*] = *max_category*[*r*] - 1

max_bits = *max_bits* + *expected_bits_table*[*max_category*[*r*]]

go to STEP(18)

STEP(16)

以下の式を満たす領域 *r* に対して

min_category[*r*] < 7

次に示す関数を最大にする領域を探索する。

offset - *rms_index*[*r*] - 2 * *min_category*[*r*]

この関数を最大にする *r* が複数存在する場合は、最も大きい（すなわち、最高周波数の）領域 *r* を選択する。

STEP(17)

categorization_count = *categorization_count* + 1

min_pointer = *min_pointer* + 1

min_bits = *min_bits* - *expected_bits_table*[*min_category*[*r*]]

min_category[*r*] = *min_category*[*r*] + 1

min_bits = *min_bits* + *expected_bits_table*[*min_category*[*r*]]

STEP(18)

categorization_count = *categorization_count* + 1

if

categorization_count < 16

then

go to STEP(13)

otherwise

go to STEP(19)

STEP(19)

max_category[]を *category.0*[]へコピーする。

すべての領域において

category.0[*r*] = *max_category*[*r*]

STEP(20)

n = 1

STEP(21)

(*n*-1)番目のカテゴリ化タイプにおける内容を *n* 番目のカテゴリ化タイプへコピーする。

すべての領域において

category.n[*r*] = *category.n*-1[*r*]

STEP(22)

temp_category_balances[*max_pointer*]によって示された領域に対するカテゴリをインクリメントする。

category.n[*temp_category_balances*[*max_pointer*]] =

category.n[*temp_category_balances*[*max_pointer*]] + 1

STEP(23)

max_pointer = *max_pointer* + 1

n = *n* + 1

STEP(24)

if

$n < 16$

then

go to STEP(21)

otherwise

END

８．図

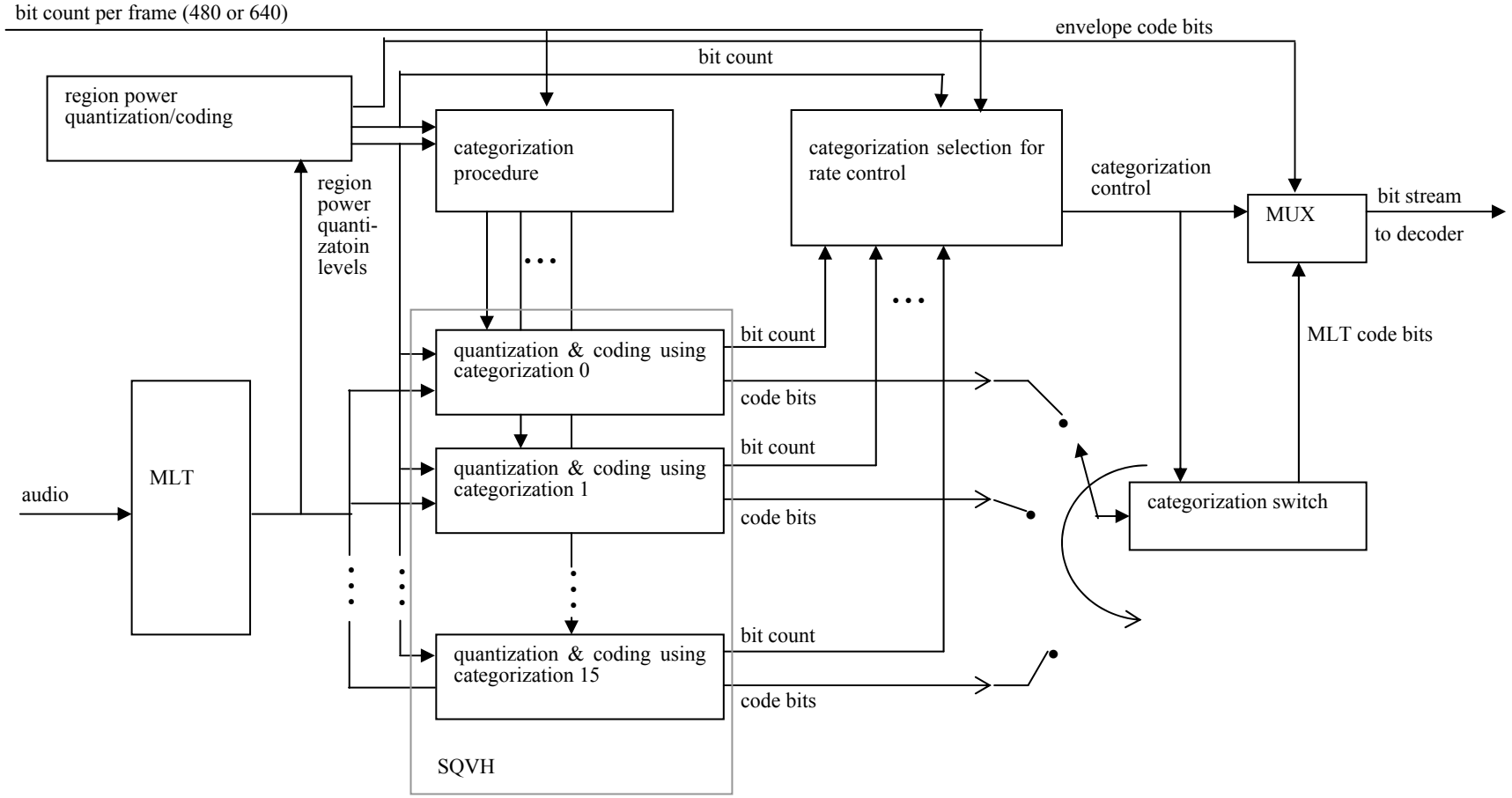


Figure 1／JT-G722.1 Block diagram of the encoder

(ITU-T G.722.1)

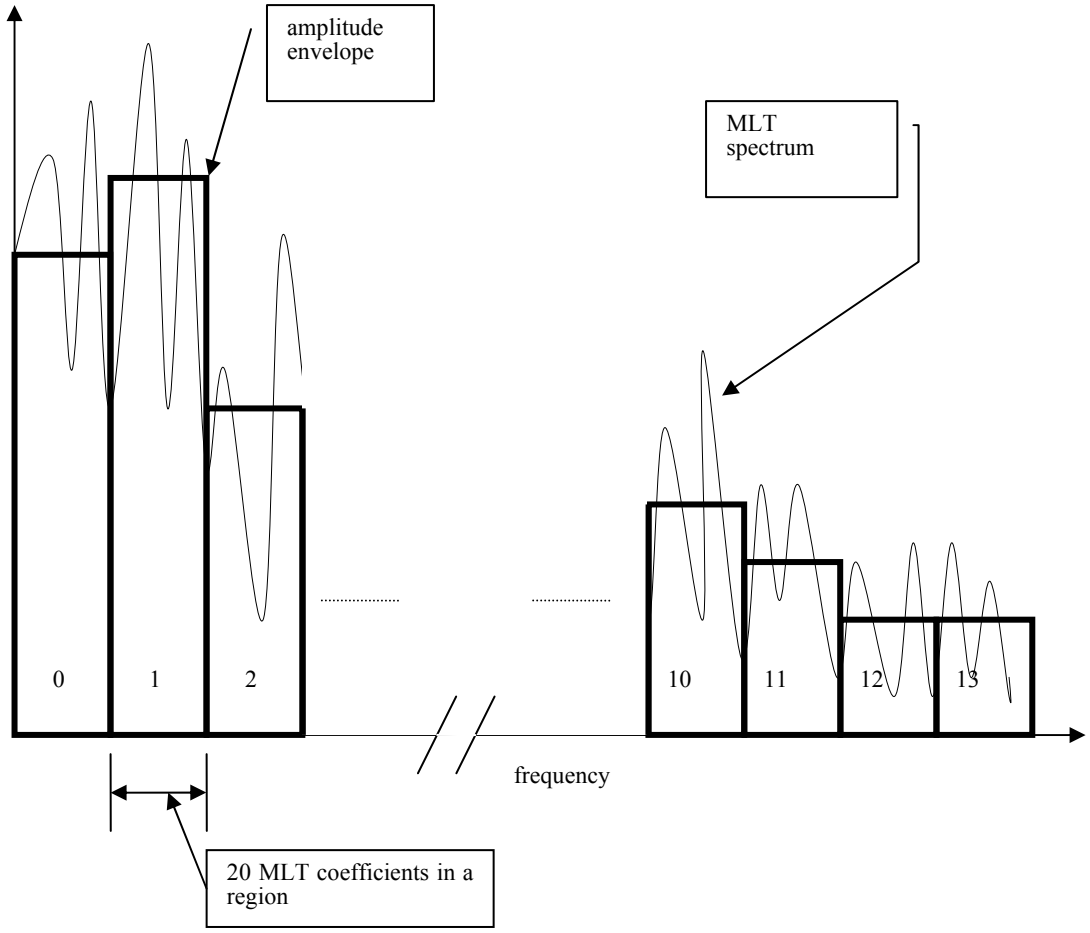


Figure 2／JT-G722.1 (ITU-T G.722.1) An illustration of how the spectrum is diveded into fourteen regions,each containing 20 MLT coefficients. Each value of the amplitude envelope represents the RMS (root-mean-square) value of the MLT coefficients in that region.

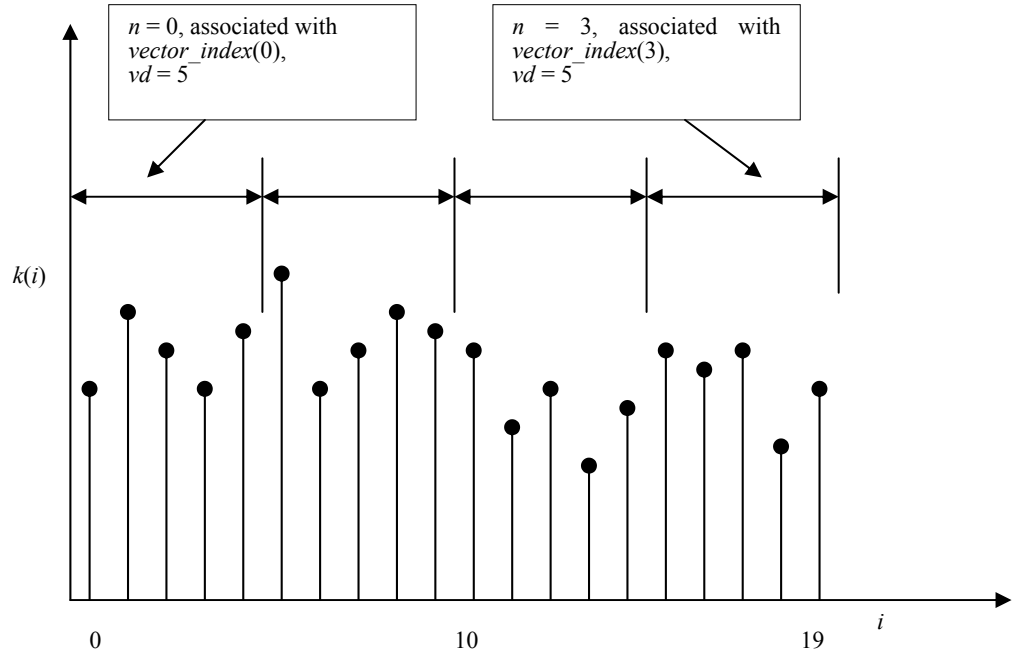


Figure 3／JT-G722.1 (ITU-T G.722.1) An example illustrating how a region, assigned a category of 6, is split into a series of five dimensional vectors ($vd = 5$) with four vectors per region ($vpr = 4$ and $0 \le n < 4$). Each vector index will represent vd quantized MLT coefficients.

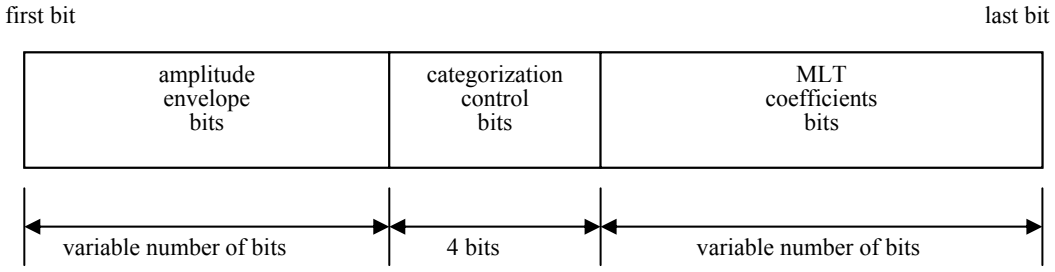


Figure 4／JT-G722.1 (ITU-T G.722.1) Major bit stream fields and their order in transmission.

付属資料Ａ

（標準ＪＴ－Ｇ７２２．１に対する）

ＪＴ－Ｇ７２２．１パケットフォーマット、能力識別子および能力パラメータ

Ａ．１　概要

本付属資料は、ＪＴ－Ｈ３２３システムで用いられる場合のＪＴ－Ｇ７２２．１に対するパケットフォーマットを記述したものである。ここで定義されたものと同一のペイロード構成は、他のパケット転送システムに対しても適用できる。ＪＴ－Ｈ３２３に必要な能力識別子とパラメータテーブルも記載される。

Ａ．２　ＪＴ－Ｇ７２２．１フレームに対するパケット構成

ＴＴＣ標準ＪＴ－Ｇ７２２．１で定義されるオーディオ符号化方式は、50Hz-7kHz 帯域幅のオーディオ信号を、フレーム長 20ms およびサンプリング周波数 16kHz で、24kbit/s または 32kbit/s のいずれかのビットレートで符号化を行う。ビットレートは、任意の 20ms フレーム単位で変更可能である。ただし、ビットレートの変更は、ビット列中のインバンドにおいては通知されない。24kbit/s で動作する場合は、１フレームあたり 480 ビット（60 オクテット）が生成され、32kbit/s で動作する場合は、１フレームあたり 640 ビット（80 オクテット）が生成される。このように、両方のビットレートは、パディングビットなしにオクテット単位の配列が可能である。

同様に、ＴＴＣ標準ＪＴ－Ｇ７２２．１付属資料Ｃにおける拡張方式は、50Hz-14kHz 帯域幅のオーディオ信号を、同じフレーム構成とパケットフォーマットで符号化する。主要なパラメータを、付表Ａ－２－１／JT-G722.1 にまとめる。

付表Ａ－２－１／JT-G722.1　ＪＴ－Ｇ７２２．１モード毎のパラメータ
(ITU-T G.722.1)

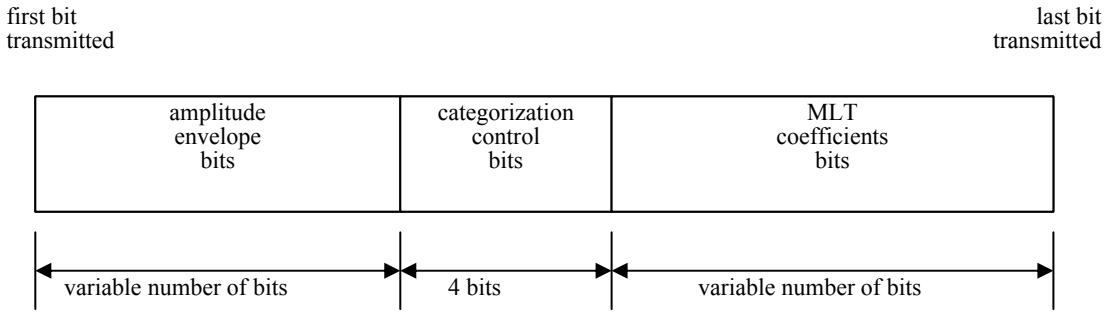
	Sample rate (kHz)	Frame length (milliseconds)	Frame size (bits/octets)
ＪＴ－Ｇ７２２．１　24 kbit/s	16	20	480 / 60
ＪＴ－Ｇ７２２．１　32 kbit/s	16	20	640 / 80
ＪＴ－Ｇ７２２．１　付属資料Ｃ　24 kbit/s	32	20	480 / 60
ＪＴ－Ｇ７２２．１　付属資料Ｃ　32 kbit/s	32	20	640 / 80
ＪＴ－Ｇ７２２．１　付属資料Ｃ　48 kbit/s	32	20	960 / 120

フレーム単位のビット数は一定である。しかしながら、この一定ビット数のフレーム内で、ＪＴ－Ｇ７２２．１では符号化パラメータの大部分を表現するのに可変長符号化（すなわち、ハフマン符号化）を用いる。カテゴリ化制御ビットパラメータを除いて、全ての他のビット列パラメータは、可変長の符号、すなわち可変ビット数で表現される。付図Ａ－２－１／JT-G722.1 ではこの点を示すと共に、送信パラメータフィールドの順序を示している。全ての可変長符号およびカテゴリ化制御ビットは最も左の（すなわち、最上位の）ビットから最も右の（すなわち、最下位の）ビットへの順で送信される。ハフマン符号化を使用するということは、すなわち、最初にフレーム全ての復号を完結しなければ、ビット列に含まれる各種パラメータ符号化パラメータ／フィールドを特定することはできないということを意味する。

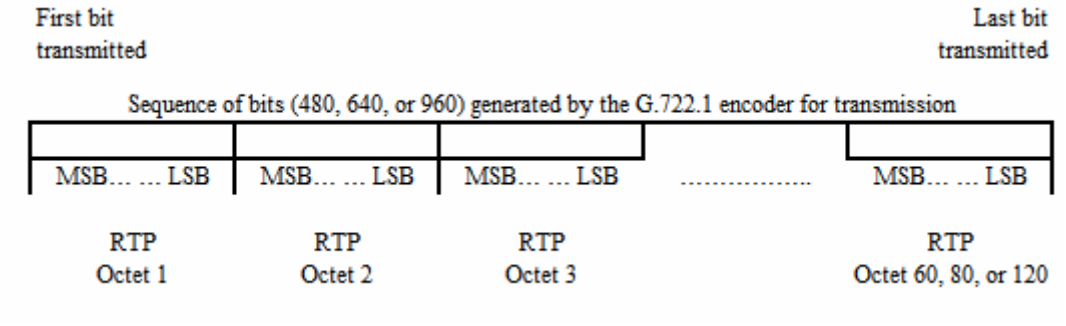
付図Ａ－２－２／JT-G722.1 に、ＴＴＣ標準ＪＴ－Ｈ２２５．０で記述されているオクテット配置されたＲＴＰ[1]ペイロードに対するＪＴ－Ｇ７２２．１のビット列のマッピング方法を図示する。

ＲＴＰパケットは同一ビットレートかつ同一サンプリング周波数のＪＴ－Ｇ７２２．１フレームのみを含むものでなければならない。ＴＴＣ標準ＪＴ－Ｇ７２２．１付属資料Ｃ以外のＲＴＰタイムスタンプは 1/16000 秒単位でなければならず、ＴＴＣ標準ＪＴ－Ｇ７２２．１付属資料Ｃの場合は、1/32000 秒単位でなければなら

ない。



付図Ａ－２－１／JT-G722.1　ビット列フィールドとその送信順序
(ITU-T G.722.1)



付図Ａ－２－２／JT-G722.1　ＪＴ－Ｇ７２２．１符号化ビット列
(ITU-T G.722.1)

このビット列は、オクテットのシーケンスに分割され（モードに応じて６０、８０、または１２０オクテット）、各々のオクテットはＲＴＰオクテットにマッピングされる。

Ａ．３　ＴＴＣ標準ＪＴ－Ｈ２４５で用いられる能力識別子およびパラメータテーブル

ＪＴ－Ｈ２４５においてベースラインＪＴ－Ｇ７２２．１の能力情報交換用に GenericCapability が使用される。能力識別子およびパラメータのテーブルを以下に示す。

Ａ．３．１　ベースラインＪＴ－Ｇ７２２．１

付表Ａ－３－１－１／JT-G722.1　ＪＴ－Ｇ７２２．１に対する能力識別子テーブル
(ITU-T G.722.1)

能力名称	ＴＴＣ標準ＪＴ－Ｇ７２２．１
能力クラス	Audio
能力識別子タイプ	標準
能力識別子値	{ itu-t (0) recommendation (0) g (7) 7221 generic-capabilities (1) 0 }
maxBitRate	このパラメータは32000（32kbit/sを表す）または24000（24kbit/sを表す）の値に設定されなければならない.
collapsing	このフィールドは、以下に与えられるＪＴ－Ｇ７２２．１Capability Parameters を含まなければならない
nonCollapsing	このフィールドは含まれてはならない
nonCollapsingRaw	このフィールドは使用されない
transport	このフィールドは使用されない

ＪＴ－Ｈ２４５能力と OpenLogicalChannel メッセージの両方において、付表Ａ－３－１－１で与えられる maxBitRate フィールドは、一つの正確なビットレートで動作する単一のＪＴ－Ｇ７２２．１モードを示すために使われなければならない（ＪＴ－Ｈ２４５における maxBitRate の定義にかかわらず）。

例えば、32000 の maxBitRate を持つ{ itu-t (0) recommendation (0) g (7) 7221 generic-capabilities (1) 0 }の能力識別子を含むシステムは、32kbit/s のみのＪＴ－Ｇ７２２．１に準じた動作能力があることを示し、いかなる低ビットレートでの動作能力を必要としない。もし、システムが 24kbit/s と 32kbit/s の両方で動作する能力がある場合、二つの GenericCapability メッセージ、すなわち一つは 24000 の maxBitRate を示し、もう一方は 32000 の maxBitRate を示す、を持つ能力を表示しなければならない。

注：付表Ａ－３－１－１における maxBitRate フィールドの単位およびその使用法は、ＪＴ－Ｈ２４５の規定とも、付表Ａ－３－２－１における単位および使用法とも異なる。これらの単位および使用法は、歴史的に確立されており、既存システムとの相互接続性を継続するために踏襲される。

付表Ａ－３－１－２／JT-G722.1　１つの RTP パケットに許容される最大フレーム数を記述した、
(ITU-T G.722.1)　ＪＴ－Ｇ７２２．１に対する GenericCapability パラメータテーブル

パラメータ名称	maxFramesPerPacket
パラメータ記述	これは Collapsing GenericParameter である. maxFramesPerPacket の値は１つの RTP パケットに含まれる最大のＪＴ－Ｇ７２２．１符号化フレーム数を示す
パラメータ識別子値	1
パラメータステータス	必須
パラメータタイプ	unsignedMin
Supercedes	このフィールドは使用されない

Ａ．３．２　ＪＴ－Ｇ７２２．１の拡張モード

これらの表は、ＴＴＣ標準ＪＴ－Ｇ７２２．１付属資料Ｃを含むＪＴ－Ｇ７２２．１の拡張モードに対するＪＴ－Ｈ２４５において使用される GenericCapability を規定している。

付表Ａ－３－２－１／JT-G722.1　ＪＴ－Ｇ７２２．１の拡張モードに対する能力識別子テーブル
(ITU-T G.722.1)

能力名称	ＴＴＣ標準ＪＴ－Ｇ７２２．１
能力クラス	Audio
能力識別子タイプ	標準
能力識別子値	{ itu-t (0) recommendation (0) g (7) 7221 generic-capabilities (1) extension (1) 0 }
maxBitRate	このパラメータは480（48kbit/sを表す）、320（32kbit/sを表す）または240（24kbit/sを表す）の値に設定されなければならない
collapsing	このフィールドは、以下に与えられるＪＴ－Ｇ７２２．１拡張モードCapability Parametersを含まなければならない
nonCollapsing	このフィールドは含まれてはならない
NonCollapsingRaw	このフィールドは使用されない
Transport	このフィールドは使用されない

この能力識別子の maxBitRate フィールドは、ＪＴ－Ｈ２４５にしたがって使用される。下記の supportedExtendedModes パラメータにおいて示されるモードのセットから、サポートされる最大ビットレートを示す。単一の{ itu-t (0) recommendation (0) g (7) 7221 generic-capabilities (1) extension (1) 0 }能力識別子は、全てのサポートされているＪＴ－Ｇ７２２．１拡張モードを示すために使われなければならない。

付表Ａ－３－２－２／JT-G722.1　ＪＴ－Ｇ７２２．１の拡張モードに対するGeneric Capabilityパラメータ表（ＲＴＰパケットにおいて許容される最大フレーム数の記述）
(ITU-T G.722.1)

パラメータ名称	maxFramesPerPacket
パラメータ記述	これは、Collapsing GenericParameterである。maxFramesPerPacketの値は、単一のRTPパケットに含まれる拡張モードＪＴ－Ｇ７２２．１符号化フレーム数の最大値を指定する。
パラメータ識別子値	1
パラメータステータス	必須
パラメータタイプ	unsignedMin
Supercedes	このフィールドは使用されない

付表Ａ－３－２－３／JT-G722.1　ＪＴ－Ｇ７２２．１の拡張モードに対する Generic Capability パラメータ表
（サポートしているモードの記述）

(ITU-T G.722.1)

パラメータ名称	supportedExtendedModes
パラメータ記述	このパラメータはブール配列である bit 2 (value 64) が1の時、24 kbit/sの付属資料Ｃ／JT-G722.1を示す bit 3 (value 32) が1の時、32 kbit/sの付属資料Ｃ／JT-G722.1を示す bit 4 (value 16) が1の時、48 kbit/sの付属資料Ｃ／JT-G722.1を示す その他の全てのビットは予約され、０に設定されなければならず、受信側では無視しなければならない それぞれのビットが１に設定された場合、システムが、示されたモードで動作する能力があることを意味する。 OpenLogicalChannelメッセージにおいては、一つのビットのみ１に設定されなければならない。 これは、論理チャネルにおいて使用されるモードを示している。 注 – 将来において、予約ビット数で提供することのできる数より多くのモードが定義された場合、追加モードに対応したほかのパラメータを割り当てることによって追加モードが示されることになる
パラメータ識別子値	2
パラメータステータス	必須
パラメータタイプ	booleanArray
Supercedes	このフィールドは使用されない

Ａ．４　参考文献

[1]　ＴＴＣ標準ＪＴ－Ｈ２２５．０

付属資料Ｂ

（標準ＪＴ－Ｇ７２２．１に対する）

ＪＴ－Ｇ７２２．１の浮動小数点演算による実現

Ｂ．１　概要

本付属資料は、ＴＴＣ標準ＪＴ－Ｇ７２２．１の浮動小数点演算による実現に関して記述したものである。

Ｂ．２　アルゴリズムの記述

ＴＴＣ標準ＪＴ－Ｇ７２２．１浮動小数点演算版における基本的なアルゴリズム手順は、固定小数点演算版と同じである。

Ｂ．３　ＡＮＳＩ Ｃコード

ＴＴＣ標準ＪＴ－Ｇ７２２．１の浮動小数点演算をシミュレートするＡＮＳＩ準拠のＣコードは、ＩＴＵ－Ｔの Web サイトから入手可能である。Ｃコードによるアルゴリズム記述は、ＴＴＣ標準ＪＴ－Ｇ７２２．１の本文および本付属資料に記載される内容より優先される。

浮動小数点演算のＣソースコードを構成するファイル一覧を TableＢ－３－１／JT-G722.1 に示す。

TableＢ－３－１／JT-G722.1　List of software files specific to G.722.1 floating-point source code (ITU-T G.722.1)

File Name	Description
common.c	routines used by encoder and decoder
dct4.c	forward and inverse DCT
decode.c	main program for decoder
decoder.c	routines for decoder
encode.c	main program for encoder
encoder.c	routines for encoder
rmlt_coefs_to_samples.c	inverse MLT
samples_to_rmlt_coefs.c	MLT
defs.h	parameter definitions
huff_defs.h	definitions for Huffman coding
huff_tables.h	declaration of Huffman tables

この浮動小数点演算プログラムをコンパイルすると､符号器ファイル encode と復号器ファイル decode が生成され、これらは次のようにコマンドラインで使えるようになる。

encode　bit-stream-type　input-audio-file　output-bit-stream-file　bit-rate

decode　bit-stream-type　input-bit-stream-file　output-audio-file　bit-rate

ここで

bit-stream-type　＝0, 圧縮されたビット列を指定

bit-stream-type　＝1, テスト用のＩＴＵ－Ｔ勧告Ｇ．１９２のビット列フォーマットを指定

input-audio-file＝ サンプルデータとして読み込ませる 16 ビットＰＣＭオーディオファイル名

output-audio-file＝ 復号出力を保存する 16 ビットＰＣＭオーディオファイル名

input-bit-stream-file = 入力ビット列として読み込ませるファイル名
output-bit-stream-file = 符号化ビット列出力を保存するファイル名
bit-rate = 24000 あるいは 32000, それぞれ 24kbit/s、32kbit/s 動作に対応

付属資料Ｃ

（標準ＪＴ－Ｇ７２２．１に対する）

２４，３２，および４８ｋｂｉｔ／ｓの１４ｋＨｚモード

概要

ＴＴＣ標準ＪＴ－Ｇ７２２．１は、16kHz の サンプリング周波数に基づく低演算量 7kHz オーディオ帯域モードについて述べている。本付属資料は、32kHz の サンプリング周波数を用いた 14kHz オーディオ帯域信号を 24,32,48kbit/s のビットレートで符号化するＴＴＣ標準ＪＴ－Ｇ７２２．１の低演算量拡張モードについて述べている。

このモードは、ビデオ会議、電話会議、およびインターネットストリーミングを用いたアプリケーションでの利用に適しており、7kHz モードと同じ 20ms のフレーム長、40ms のアルゴリズム遅延、および同じアルゴリズム上のステップを用いる。7kHz モードでは、符号化および復号に 5.5WMOPS 未満の演算量が必要であるのに対し、14kHz モードでは、符号化および復号に 11 WMOPS 未満の演算量が必要である。

24,32,および 48kbit/s の 14kHz モードのＴＴＣ標準ＪＴ－Ｇ７２２．１付属資料Ｃの固定小数点リファレンスコードは、ＩＴＵ－Ｔの Web サイトから入手可能である。

Ｃ．１　はじめに

本付属資料は、ＴＴＣ標準ＪＴ－Ｇ７２２．１の 24,32,および 48kbit/s の 14kHz モードの詳細を提供するものである。

Ｃ．２　アルゴリズムの記述

このＴＴＣ標準ＪＴ－Ｇ７２２．１の 14kHz モードは、二倍になったオーディオ帯域に適応させるためにアルゴリズム上の一部定数が二倍になったことを除き、ＴＴＣ標準ＪＴ－Ｇ７２２．１本体のモード（7kHz オーディオ帯域）と同一のアルゴリズム上のステップを持つ。

ＴＴＣ標準ＪＴ－Ｇ７２２．１本体と比較した本付属資料のアルゴリズム上の相違点を明示すると、以下の通り。

ａ）サンプリングレートが 16kHz から 32kHz と二倍になっている。

ｂ）フレームあたりのサンプル数が 320 から 640 と二倍になっている。（同じ 20ms のフレーム長のため）

ｃ）変換窓サイズが以下のように 640 から 1280 と二倍になっている。

それぞれのＭＬＴの入力は、最近の 1280 オーディオサンプル $x(n)$である。

ここで、$x(0)$が最も古いサンプル値であり、$0 \le n < 1280$ である。

ＭＬＴの出力は、640 個の変換係数 $mlt(m)$である。

ここで、$0 \le m < 640$ である。

ＭＬＴは以下のように与えられる。

$$mlt(m) = \sum_{n=0}^{1279} \sqrt{\frac{2}{640}} \sin\left(\frac{\pi}{1280}(n+0.5)\right) \cos\left(\frac{\pi}{640}(n-319.5)(m+0.5)\right) x(n)$$

ＭＬＴは、窓掛け重ね合せ加算演算と、引き続き行われるタイプⅣの離散コサイン変換（ＤＣＴ）とに分解することができる。窓掛け重ね合せ加算演算は、以下のように与えられる。

$$v(n) = w(319-n)x(319-n) + w(320+n)x(320+n) \quad \text{for } 0 \le n \le 319$$

$$v(n+320) = w(639-n)x(640+n) - w(n)x(1279-n) \quad \text{for } 0 \le n \le 319$$

ここで、

$$w(n) = \sin\left(\frac{\pi}{1280}(n+0.5)\right) \quad \text{for } 0 \le n < 640$$

である。

$v(n)$をタイプⅣのＤＣＴと結合することにより、ＭＬＴの最終的な表現はこのようになる。

$$mlt(m) = \sum_{n=0}^{639}\sqrt{\frac{2}{640}}\cos\left(\frac{\pi}{640}(n+0.5)(m+0.5)\right)v(n) \qquad \text{for } 0 \le m < 640$$

それぞれのＩＭＬＴ演算は、640 個の係数から 640 個の時間領域オーディオサンプルを生成する。ＩＭＬＴは、タイプⅣのＤＣＴと、引き続き行われる窓掛けオーバーラップ加算演算とに分解することができる。

タイプⅣのＤＣＴは、以下のように与えられる。

$$u(n) = \sum_{m=0}^{639}\sqrt{\frac{2}{640}}\cos\left(\frac{\pi}{640}(m+0.5)(n+0.5)\right)mlt(m)$$

窓掛けオーバーラップ加算演算は、以下のように、現在のフレームのＤＣＴ出力の半分のサンプルと、前のフレームのＤＣＴ出力の半分のサンプルとを用いる。

$$y(n) = w(n)u(319-n) + w(639-n)u\_old(n) \qquad \text{for } 0 \le n \le 319$$

$$y(n+320) = w(320+n)u(n) - w(319-n)u\_old(319-n) \qquad \text{for } 0 \le n \le 319,$$

ここで、

$$w(n) = \sin\left(\frac{\pi}{1280}(n+0.5)\right) \qquad \text{for } 0 \le n \le 639$$

である。

ここで用いられなかった $u()$の半分は、次のフレームで用いるために以下のように $u_old()$として蓄積される。

$$u\_old(n) = u(n+320) \qquad \text{for } 0 \le n \le 319$$

ｄ）以下のように、二倍になった帯域に適応させるために、新たにＤＣＴテーブルを追加している。

ＤＣＴのために新たなテーブル *a_cos_msin_64[320][2]*が *dct4_a.h* に、また逆ＤＣＴのために二つの新たなテーブル *s_cos_msin_64[320][2]*および *max_dither[640]*が *dct4_s.h* に、それぞれ追加されている（詳細は *dct4_a.h* および *dct4_s.h* を参照すること）。

ｅ）500Hz 幅のサブバンドの数を、14 から 28 へ二倍している。

ｆ）以下のように、ハフマン符号化テーブルを二倍している。

以下のハフマン符号化テーブルは、ＴＴＣ標準ＪＴ－Ｇ７２２．１本体で用いられているハフマン符号化テーブルと比べてより大きなサイズとなっている。

differential_region_power_bits[28][24]

differential_region_power_codes[28][24]

differential_region_power_decoder_tree[28][23][2]

mlt_quant_centroid[8][16]

上記の最初の 3 つのテーブルは、ＴＴＣ標準ＪＴ－Ｇ７２２．１本体用のそれぞれ対応するテーブルの最終行を繰り返すことによって得られる。4 番目のテーブルは、ＴＴＣ標準ＪＴ－Ｇ７２２．１本体用のテーブルに、新たに 1 行と零の列 2 列分を加えることによって得られる（詳細は、huff_tab.c を参照すること）。

ｇ）1 フレーム当たりの合計ビット数は、ビットレートが 24、32、及び 48 kbit/s において、それぞれ 480、640、及び 960 ビットとなる。これには、振幅包絡を表すために使われるビット、4 つのカテゴリ化制御ビット、及び MLT 係数符号化のために必要とされるビットが含まれる。ビット割り当ては、フレーム毎に、14kHz の全周

波数帯域に渡って動的に変化させる。

ｈ）以下のように、使用可能なビット数を調整するためのしきい値が二倍となっている。
実際の使用可能なビット数に基づいて、次式にて使用可能なビット数の推定値を計算する。
if

$$number\_of\_available\_bits > 640$$

then

$$estimated\_number\_of\_available\_bits = 640 + ((number\_of\_available\_bits - 640) * 5/8)$$

estimated_number_of_available_bits は、カテゴリ化の過程において、常に実際の使用可能なビット数未満であり、余裕を持たせている。

Ｃ．３　ＡＮＳＩ　Ｃコード

ＴＴＣ標準ＪＴ－Ｇ７２２．１の 14 kHz モードをシミュレートするＡＮＳＩ準拠のＣコードは、ＩＴＵ－Ｔの Web サイトから入手可能である。Ｃコードによるアルゴリズム記述は、ＴＴＣ標準ＪＴ－Ｇ７２２．１本体または本付属資料に記載される内容より優先される。

16/32 ビット固定小数点演算のＣソースコードを構成するファイル一覧を、TableＣ－３－１／JT-G722.1 に示す。

TableＣ－３－１／JT-G722.1　List of software files specific to G.722.1 14 kHz mode source code (ITU-T G.722.1)

File Name	Description
basop32.c	basic arithmetic operators
coef2sam.c	inverse MLT
common.c	routines used by encoder and decoder
count.c	functions for automatic complexity calculation
dct4_a.c	forward DCT
dct4_s.c	inverse DCT
decode.c	main program for decoder
decoder.c	routines for decoder
encode.c	main program for encoder
encoder.c	routines for encoder
huff_tab.c	Huffman coding for both encoder & decoder
sam2coef.c	forward MLT
tables.c	tables for forward & inverse MLT
basop32.h	definitions of basic arithmetic operators
count.h	definitions of functions for measuring complexity
dct4_a.h	definitions of tables for forward DCT
dct4_s.h	definitions of tables for inverse DCT
defs.h	parameter definitions
huff_defs.h	definitions for Huffman coding
huff_tables.h	declaration of Huffman tables

tables.h	definitions of tables for forward & inverse MLT
typedefs.h	definitions of data types and constants

これら個々のプログラムをコンパイルすると、符号器ファイル encode と復号器ファイル decode が生成され、これらは以下のようにコマンドラインで使えるようになる。

encode　bit-stream-type　input-audio-file　output-bit-stream-file　bit-rate　bandwidth

decode　bit-stream-type　input-bit-stream-file　output-audio-file　bit-rate　bandwidth

ここで、

bit-stream-type　= 0,　圧縮されたビット列を指定

bit-stream-type　= 1,　テスト用のＩＴＵ－Ｔ勧告Ｇ．１９２のビット列フォーマットを指定

input-audio-file = サンプルデータとして読み込ませる 16 ビットＰＣＭオーディオファイル名

output-bit-stream-file = 符号化ビット列出力を保存するファイル名

input-bit-stream-file = 入力ビット列として読み込ませるファイル名

output-audio-file = 復号出力を保存する 16 ビットＰＣＭオーディオファイル名

bit-rate = 24000 あるいは 32000,　7kHz オーディオ帯域モードのビットレート(単位は bit/s)

bit-rate = 24000, 32000 あるいは 48000,　14kHz オーディオ帯域モードのビットレート(単位は bit/s)

bandwidth = 7000 あるいは 14000,　それぞれ 7 kHz オーディオ（ＴＴＣ標準ＪＴ－Ｇ７２２．１本体）、あるいは 14 kHz オーディオ（ＴＴＣ標準ＪＴ－Ｇ７２２．１付属資料Ｃ）に対応

付録

（標準ＪＴ－Ｇ７２２．１に対する）

用語対照表

英語	ＴＴＣ標準用語
2's complement	２の補数
amplitude envelope	振幅包絡
assignment	割当て
audio	オーディオ
bandwidth	帯域幅
basis function	基底関数
bit exact	ビットイグザクト
bit stream	ビット列
capability exchange	能力情報交換
capability identifier	能力識別子
categorization	カテゴリ化タイプ
categorization control bit	カテゴリ化制御ビット
categorization procedure	カテゴリ化手順
category	カテゴリ
category assignment	カテゴリ割り当て
category table	カテゴリテーブル
category value	カテゴリ値
centroid table	セントロイドテーブル
code bit	符号語ビット
coder	符号器
decoder	復号器
discrete cosine transform	離散コサイン変換
encoder	符号器
fixed-point mathematical operations	固定小数点演算
floating point	浮動小数点
frame loss	フレーム消失
hands free	ハンズフリー
Huffman coding	ハフマン符号化
inband	インバンド
insufficient bits	不足ビット
least significant bit	最下位ビット
look-ahead	先読み
modulated lapped transform	変調重複変換
most significant bit	最上位ビット
multiplexer	多重化器
noise fill	雑音充填
octet	オクテット
overlap and add operation	重ね合わせ加算演算

英語	ＴＴＣ標準用語
packet format	パケットフォーマット
packet transport	パケット転送
padding	パディング
payload	ペイロード
pseudo-random number	擬似乱数
region	領域
scalar quantization	スカラ量子化
sign	符号
sign bit	符号ビット
spectrum	スペクトル
time stamp	タイムスタンプ
variable length	可変長
wideband	広帯域

付録

（標準ＪＴ－Ｇ７２２．１に対する）

用語解説

離散コサイン変換（discrete cosine transform）

余弦関数から導かれる基底ベクトルによる直交変換の一種であり、高速変換アルゴリズムが存在するなどの実現上の利点がある変換手法。

ハフマン符号化（Huffman coding）

量子化された事象に対して、その発生頻度により長さの異なる符号を割り当てることによりデータ圧縮を行う手法である可変長符号化の中で代表的な符号化法。ハフマン符号では、事象の発生頻度を予め想定しておき、それに対して可変長符号が設計される。

変調重複変換（modulated lapped transform）

直交変換手法のうち、隣接ブロックとデータを重複させて信号の変換を行う方式である、重複直交変換手法のうちの１手法。